KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Trustworthy kernel machines: diversity, robustness, and disentanglement

Joachim Schreurs

Supervisor:
Prof. dr. ir. Johan A.K. Suykens

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

October 2021

Trustworthy kernel machines: diversity, robustness, and disentanglement

Joachim SCHREURS

Examination committee:
Prof. dr. Adhemar Bultheel, chair
Prof. dr. ir. Johan A.K. Suykens, supervisor
Prof. dr. ir. Panos Patrinos
Prof. dr. Jesse Davis
Prof. dr. Carolina Varon Perez
(Université Libre de Bruxelles)
Prof. dr. Michael Mahoney
(UC Berkeley)
Prof. dr. ir. Benoît Frénay
(Université de Namur)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

October 2021

© 2021 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Joachim Schreurs, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Preface

This text is the result of four years of research as a PhD student at the department of electrical engineering of KU Leuven under the supervision of Prof. Johan Suykens. This text would have never been possible without the help and support of many other people. Every single one of them has my sincerest gratitude.

First and foremost, I would like to thank professor Johan Suykens, my supervisor, for the many opportunities he has given me, the invaluable insights, and the interesting discussions. Your encouragement and excitement for research are something I will never forget.

Next, I would like to thank all members of the examination committee for reading this thesis and your helpful feedback: Prof. Adhemar Bultheel, Prof. Panos Patrinos, Prof. Jesse Davis, Prof. Carolina Varon Perez, Prof. Michael Mahoney, and Prof. Benoît Frénay.

I would like to thank my current and former colleagues for the pleasant environment: Yingyi, Bram, Henri, Michaël, Lynn, Zahra, Fanghui, Arun, Soutrik, Qinghua, Francesco, Hannes, Mauricio, Kostas, and David. Thank you all for making these last years even more special. We had some amazing times during coffee breaks, lunches, drinking beers, and many ping-pong breaks. I would also like to thank the administrative and technical staff at STADIUS for helping me with all the bureaucratic and technical issues that I faced along the way.

A few people deserve special thanks. Lynn and Zahra, you welcomed me into the group when I first joined. Thank you for helping me to get started, you were always there when I had questions. I very much enjoyed the beers that we had together, during the weekly-scheduled beer time, or at the Oude Markt watching football.

Michaël, you truly were my role model during the PhD. You inspired me to always reach for the best, be thorough and precise, and most of all enjoy doing research. I always enjoyed discussing with you, whether it be on difficult research topics or rambling on the latest developments in Belgian politics. This thesis would not have been possible without your help.

Arun, thank you for the many years of interesting research. I will always remember our conference together in Italy, a little bit smaller as expected due to Covid, but equally as enjoyable. We still have to sample some Italian wines together.

Hannes, we first met as teaching assistants for the course on 'Artificial neural networks and deep learning'. We directly hit it off and started collaborating together. First at the office, afterwards over our many Zoom meetings. I can not count the number of times you helped me with yet another Tensorflow problem. I am really glad to have met you, and to call you my friend.

Lastly, special thanks go out to Iwein. Thank you for introducing me to the field of robust statistics. You really sparked my interest on that faithful day during the course of 'managing your PhD', perhaps the most important thing I learned that day. I will miss our lunches together at Alma and our many (late) phone calls.

You all are amazing persons and I will truly miss working with all of you.

A special thanks goes to Celus, Dries, Maarten, Mitch, Olivier, Stijn, Tim, Wouter, Christof, and Jeroen. I want to thank you for reminding me that there is life outside of research. Thank you for the skiing trips, the festivals, the parties, the dinners, the padel matches, and so much more. I am proud to call such an amazing group of people my friends.

I owe special gratitude to my family. Omie, thank you for your unconditional support and love. Mama and papa, you helped me become the person I am today. Thank you for always being there for me, I dedicate this manuscript to you. Hannah and Lucas, thank you for all the nice moments we had together. Remember to always stay true to yourself, do what you love, and everything will work out. The same holds for Dieter and Margaux, I am happy you joined our little family. Special thanks go out to my family-in-law, thank you for all the nice talks, the many drinks, and delicious barbeques. Finally, I would like to thank Yentl. The last four years were not always easy, but coming back to you gave me courage. Thank you for putting a smile on my face when I most needed it, for your love, for your patience, for everything.

Abstract

Machine learning models are used more and more in critical applications such as healthcare, credit scoring, or justice. As these technologies play a more prominent role in our everyday lives, it is important to ask ourselves some questions. Models can be biased when certain subpopulations are not well represented in the data. Is it still possible to build trustworthy algorithms that remove potential biases and provide a fair prediction for all groups? To establish trust in models, can we provide some interpretable information to the user? When training data is noisy, can we trust the algorithm to not be affected and still make correct predictions? In this thesis, we try to answer and provide a solution to these questions.

The first question is related to fairness and bias in machine learning. Datasets often contain a mixture of different subpopulations where the frequency of each subpopulation can be vastly different. When training machine learning models on these datasets, it is important to not only predict well in the majority of the data but also for specific outlying points. When using sampling algorithms to train these models, it is therefore crucial to select points out of each subpopulation to achieve close-to-optimal generalization error. In this thesis, we propose diversity sampling to increases the probability of every subpopulation being included in each training batch, hence reducing potential bias in the model.

When working with generative models, it is often beneficial to provide some interpretable information to the user to gain trust in the model. A considerable number of works on interpretability in representation learning and generative modeling focuses on disentanglement, which measures how well individual components in the latent space match ground-truth factors that generated the data. Because these components are interpretable, it is possible to carefully guide the generation process to the desired outcome. In this thesis, we propose generative restricted kernel machines. A generative model that lies at the intersection between kernel methods and neural networks. By including a kernel

PCA loss in the generative model, we get an orthogonal latent space, which in turn leads to disentanglement.

Finally, real-world datasets often contain some unwanted observations or outliers caused by human error, a change in system behavior, or faulty sensors. When training machine learning models, it is important to detect these outliers as they can have a harmful effect on the performance and the conclusions drawn from the data. This brings us to robustness against outliers, the final theme of this thesis. Two robust methods are proposed: a robust generative method using the restricted kernel machine framework, and a kernel-based outlier detection method that follows from the generalization of the minimum covariance determinant estimator.

Beknopte samenvatting

Machine learning modellen worden steeds meer gebruikt in kritieke toepassingen zoals gezondheidszorg, crediet beoordeling of justitie. Aangezien deze technologieën een meer prominente rol spelen in ons dagelijks leven, is het belangrijk om onszelf enkele vragen te stellen. Modellen kunnen vertekend zijn wanneer bepaalde subpopulaties niet goed vertegenwoordigd zijn in de data. Is het nog steeds mogelijk om betrouwbare algoritmen te trainen die potentiële vooroordelen wegnemen en een eerlijke voorspelling voor alle groepen bieden? Kunnen we de gebruiker interpreteerbare informatie geven om vertrouwen in modellen te vergroten? Als trainingsgegevens uitschieters bevatten, kunnen we er dan van uit gaan dat het algoritme niet wordt beïnvloed en toch correcte voorspellingen doet? In dit proefschrift proberen we deze vragen te beantwoorden.

De eerste vraag heeft betrekking op eerlijkheid en vooroordelen in machine learning. Datasets bevatten vaak verschillende subpopulaties waarbij de frequentie van elke subpopulatie enorm kan verschillen. Bij het trainen van machine learning modellen op deze datasets is het belangrijk om niet alleen een goede voorspelling te maken over het meerdendeel van de data, maar ook voor specifieke afgelegen punten. Bij het gebruik van samplingalgoritmen om deze modellen te trainen, is het daarom belangrijk om punten uit elke subpopulatie te selecteren om een bijna optimale generalisatiefout te bereiken. In dit proefschrift stellen we een diversiteitssteekproef voor om de kans te vergroten dat elke subpopulatie in elke trainingsbatch wordt opgenomen, waardoor mogelijke vertekening in het model wordt verminderd.

Tijdens het gebruik van generatieve modellen is het vaak nuttig om de gebruiker interpreteerbare informatie te verschaffen om vertrouwen in het model te krijgen. Een aanzienlijk aantal papers over interpreteerbaarheid in representatieleren en generatieve modellering richt zich op een scheiding van de onderliggende generatieve factoren die de gegevens hebben gegenereerd. Omdat elk van deze componenten interpreteerbaar is, is het mogelijk om het generatieproces

zorgvuldig naar het gewenste resultaat te leiden. In dit proefschrift stellen we generative restricted kernel machines voor, d.i. een generatief model dat op de intersectie tussen kernelmethoden en neurale netwerken ligt. Door een kernel PCA verliesfunctie op te nemen in de objectfunctie, krijgen we een orthogonale latente ruimte, wat op haar beurt leidt tot een scheiding van de onderliggende generatieve factoren.

Ten slotte bevatten datasets vaak enkele ongewenste waarnemingen of uitschieters die worden veroorzaakt door menselijke fouten, een verandering in het systeemgedrag of defecte sensoren. Bij het trainen van machine learning modellen is het belangrijk om deze uitschieters te detecteren, omdat ze een schadelijk effect kunnen hebben op de prestaties en de conclusies die uit de gegevens worden getrokken. Dit brengt ons bij robuustheid tegen uitschieters, het laatste thema van dit proefschrift. Er worden twee robuuste methoden voorgesteld: een robuuste generatieve methode die gebruik maakt van het restricted kernel machine raamwerk, en een op kernels gebaseerde uitschieterdetectiemethode die volgt uit de generalisatie van de minimale covariantiedeterminantschatter.

List of Abbreviations

C-steps	Concentration steps.
CVaR	Conditional value-at-risk.
DAS	Deterministic adaptive sampling.
DPP	Determinantal point process.
ELBO	Evidence lower bound.
EU	European Union.
FID	Fréchet inception distance.
GAN	Generative adversarial network.
GDPR	General data protection regulation.
Gen-RKM	Generative restricted kernel machine.
GMM	Gaussian mixture model.
IS	Inception score.
KL	Kullback–Leibler.
KMRCD	Kernel minimum regularized covariance determinant.
KPCA	Kernel principal component analysis.
KRR	Kernel ridge regression.
LS	Leverage scores.
LS-SVM	Least squares support vector machine.
MAPE	Mean absolute percentage error.
MCD	Minimum covariance determinant.
MD	Mahalanobis distance.

MMD	Maximum mean discrepancy.
MRCD	Minimum regularized covariance determinant.
MSE	Mean squared error.
NMI	Normalized mutual information.
PCA	Principal component analysis.
PixelRNN	Pixel recurrent neural network.
PSD	Positive semidefinite.
RAS	Randomized adaptive sampling.
RBF	Radial basis function.
RBM	Restricted Boltzmann machine.
RKHS	Reproducing kernel Hilbert space.
RKM	Restricted kernel machine.
RLS	Ridge leverage score.
RRLS	Recursive leverage score sampling method.
SDO	Stahel-Donoho outlyingness.
SM	Supplementary Material.
SMAPE	Symmetric mean absolute percentage error.
SVD	Singular value decomposition.
SVM	Support vector machine.
SWD	Sliced Wasserstein distance.
VAE	Variational autoencoder.
VSC	Vlaams supercomputer centrum.
WAE	Wasserstein auto-encoder.

List of Symbols

$\lvert \cdot \rvert$	Absolute value of matrix or scalar.
adj	Adjugate of a matrix.
argmax	Argument of the maximum.
argmin	Argument of the minimum.
$\boldsymbol{a}$	Vector.
$\lvert\mathcal{S}\rvert$	Cardinality of a set $\mathcal{S}$.
$.^\top$	Transpose.
$\circ$	The entry-wise product between matrices.
diag	Diagonal of a matrix.
grad	Gradient.
Hess	Hessian.
κ	Condition number.
$\lvert\lvert \cdot \rvert\rvert_2$	L_2 norm.
mad	Median absolute deviation.
$\mathbb{I}_n$	Identity Matrix of size $n \times n$.
$\mathbb{R}$	Real numbers.
$\mathbb{R}^n$	n-dimensional vector space over $\mathbb{R}$.
det	Determinant of a matrix.
proj	Projector.

Σ	Covariance matrix.
St	Stiefel.
supp	Supremum.
Tr	Trace of a matrix.
$\varphi(\cdot)$	Feature map.
vol	Volume.
$a \in A$	a is an element of the set A.
A	Matrix.
a	Scalar.
A^+	The Moore-Penrose pseudo-inverse of matrix A.
A_{ij}	ij-th entry of the matrix A.
$E[\cdot]$	Expectation operator.
K	Kernel matrix.
$k(\cdot, \cdot)$	Kernel function.

Contents

Abstract iii

Beknopte samenvatting v

List of Abbreviations viii

List of Symbols x

Contents xi

List of Figures xvii

List of Tables xxi

1 Introduction 1
- 1.1 With great power comes great responsibility . . . 2
- 1.2 Background information . . . 6
  - 1.2.1 The kernel trick . . . 6
  - 1.2.2 The determinant of the kernel matrix . . . 9
  - 1.2.3 Generative modeling . . . 13
- 1.3 Research aims . . . 16
- 1.4 Outline of the thesis . . . 18

2 Diversity sampling and implicit regularization 21
- 2.1 Diversity sampling is an implicit regularization for kernel methods 22
  - 2.1.1 Introduction . . . 22
  - 2.1.2 Main results: Implicit regularization . . . 26
  - 2.1.3 Experimental results . . . 30
- 2.2 Ensemble kernel methods and implicit regularization . . . 36
  - 2.2.1 Introduction . . . 36
  - 2.2.2 The analogous result for kDPP sampling . . . 38

2.2.3 Experimental results 42
2.2.4 Parameters and dataset descriptions 43
2.3 Conclusion 44

3 Diversity sampling methods 45
3.1 Nyström landmark sampling and regularized Christoffel functions 45
3.1.1 Introduction 45
3.1.2 Kernel matrix approximation and determinantal point processes 47
3.1.3 Christoffel functions and determinantal point processes 52
3.1.4 Deterministic adaptive landmark selection 56
3.1.5 Randomized adaptive landmark sampling 57
3.1.6 Numerical results 63
3.2 Towards deterministic diverse subset sampling 68
3.2.1 Introduction 68
3.2.2 Deterministic adaptation of k-DPP 70
3.2.3 Connections with DAS 70
3.2.4 Numerical results 71
3.2.5 Illustration 72
3.3 Conclusion 74

4 Diversity in generative adversarial networks 77
4.1 The Bures metric for generative adversarial networks 78
4.1.1 Introduction 78
4.1.2 Method 79
4.1.3 Empirical evaluation of mode collapse 83
4.1.4 High-quality generation using a ResNet architecture . . 91
4.2 Leverage score sampling for complete mode coverage in GANs . 92
4.2.1 Introduction 92
4.2.2 Sampling with ridge leverage scores 95
4.2.3 Numerical experiments 99
4.3 Conclusion 106

5 Disentanglement and restricted kernel machines 107
5.1 Latent space exploration using generative kernel PCA 108
5.1.1 Introduction 108
5.1.2 Kernel PCA in the RKM framework 109
5.1.3 Experiments 112
5.1.4 Novelty detection 119
5.2 Generative restricted kernel machines 121
5.2.1 Introduction 121
5.2.2 Related work 122
5.2.3 Generative restricted kernel machines framework 123

5.2.4 The proposed algorithm with implicit & explicit feature maps . . . 127
5.2.5 The Gen-RKM algorithm . . . 129
5.2.6 Experiments . . . 130
5.3 Disentangled representation learning . . . 137
5.3.1 Introduction . . . 137
5.3.2 Proposed model . . . 139
5.3.3 PCA and disentanglement . . . 142
5.3.4 Connections with the Evidence Lower Bound . . . 144
5.3.5 Experiments . . . 145
5.4 Conclusion . . . 149

6 Robustness **151**
6.1 Robust generative restricted kernel machines . . . 151
6.1.1 Introduction . . . 151
6.1.2 Weighted restricted kernel machines . . . 153
6.1.3 Robust estimation of the latent variables . . . 155
6.1.4 Experiments . . . 157
6.2 Outlier detection in non-elliptical data by kernel MRCD . . . 161
6.2.1 Introduction . . . 161
6.2.2 The MCD and MRCD methods . . . 163
6.2.3 The kernel MRCD estimator . . . 165
6.2.4 The kernel MRCD algorithm . . . 169
6.2.5 Simulation study . . . 179
6.2.6 Experiments . . . 185
6.3 Conclusion . . . 188

7 Conclusion **191**

A Diversity sampling is an implicit regularization for kernel methods **195**
A.1 Proofs . . . 195
A.2 Parameters and dataset descriptions . . . 198
A.3 Additional experiments . . . 199

B Nyström landmark sampling and regularized Christoffel functions **207**
B.1 Christoffel function and orthogonal polynomials . . . 207
B.2 Useful results and Deferred Proofs . . . 208
B.3 Proof of the results of DAS . . . 213
B.4 Proof of the results of RAS . . . 216
B.5 Ablation study for different bandwidths of the Gaussian kernel 221
B.6 Simulation details, additional tables and figures . . . 221

C The Bures metric for generative adversarial networks **225**

C.1 Details of the theoretical results . . . 225
C.2 Additional experiments . . . 226
C.2.1 Timings . . . 227
C.2.2 Best inception scores achieved with DCGAN architecture 227
C.2.3 Influence of the number of convolutional layers for DCGAN architecture . . . 228
C.3 Additional figures . . . 230
C.4 Architectures . . . 236
C.4.1 Synthetic architectures . . . 236
C.4.2 CIFAR-10 and 100 DCGAN architectures . . . 239
C.4.3 STL-10 DCGAN architectures . . . 239
C.4.4 ResNet architectures . . . 240

D Leverage score sampling for complete mode coverage in GANs 243
D.1 Extra table UNBALANCED MNIST dataset . . . 243
D.2 Effect of the regularization parameter in RLS . . . 244
D.2.1 Synthetic data . . . 244
D.2.2 Unbalanced MNIST . . . 245
D.3 Timings . . . 248
D.4 Distribution of generated samples for the artificial datasets . . 248
D.5 Architectures, algorithms, and settings . . . 249

E Generative restricted kernel machines 253
E.1 Computing latent variables using covariance matrix . . . 253
E.2 Stabilizing the objective function . . . 254
E.3 Architecture details . . . 256
E.4 Extra experiments and visualizations . . . 258
E.4.1 Bilinear interpolation . . . 258
E.4.2 Visualizing the disentanglement metric . . . 258
E.4.3 Further empirical results . . . 260
E.4.4 Illustration on toy example using a Gaussian kernel . . 261

F Disentangled representation learning 265
F.1 Theoretical results . . . 265
F.1.1 Proof of Lemma 5.3.1 . . . 265
F.1.2 Details on Evidence Lower Bound for St-KM model . . 266
F.2 Datasets, hyperparameters and algorithm . . . 267
F.3 Ablation studies . . . 268
F.4 Additional empirical results . . . 272
F.5 Additional figures . . . 273

G Outlier detection in non-elliptical data by kernel MRCD 275
G.1 Algorithms . . . 275

G.2 The special case of the linear kernel 278
G.3 Additional experiments 278
G.3.1 Effect of the imposed condition number 278
G.3.2 Comparison of initial estimators 279
G.3.3 Additional simulation results with linear kernel 281
G.4 Additional figures for non-elliptical data 284

Bibliography **285**

Curriculum **309**

List of Figures

1.1.1 Illustration of bias in data. 4
1.2.1 Illustration of the kernel trick. 7
1.2.2 Illustration of DPP sampling. 11
1.2.3 Illustration MCD 13
1.2.4 Illustration GAN 14
1.2.5 Illustration VAE 15
1.2.6 Illustration of disentanglement. 16
1.4.1 Overview of the thesis 18

2.1.1 Nyström approximation for the `Housing` dataset. 23
2.1.2 Toy example of regression. 26
2.1.3 Large-scale Nyström approximation results. 32
2.1.4 KPCA results. 32
2.1.5 Large-scale KPCA results. 33
2.1.6 Regression results. 34
2.1.7 Regression results. 35
2.1.8 Large-scale Regression results. 35
2.1.9 Preconditioning results. 36
2.2.1 Ensemble KRR on the Abalone and Wine Quality dataset. . . . 38
2.2.2 Ensemble KRR on the Bikesharing and CASP dataset. 40
2.2.3 Ensemble Nyström approximation on the Adult dataset. 43

3.1.1 Samplings of an artificial data set. 46
3.1.2 Illustration Christoffel function. 55
3.1.3 Relative operator norm of the Nyström approximation error as a function of the number of landmarks. 64
3.1.4 Boxplot of the Nyström approximation error in Frobenius norm for `MiniBooNE`. 65
3.1.5 Boxplot of the Nyström approximation error in Frobenius norm. 66
3.1.6 Boxplot of the logdet($K_{\mathcal{CC}}$), SMAPE in the bulk and tail. 67

3.2.1 Illustration of sampling methods on an artificial dataset. 71
3.2.2 Singular value spectrum of the datasets on a logarithmic scale. . 72
3.2.3 Timings for the computations of Figure 3.2.6 as a function of the number of landmarks. 73
3.2.4 Illustration of the proposed method with $k = 4$ on the `Stanford Dogs` dataset. 73
3.2.5 Illustration of the proposed method with $k = 9$ on the `Kimia99` dataset. 74
3.2.6 $\log(\det(K_{CC}))$, relative operator norm, and relative max norm in function of the number of landmarks. 76

4.1.1 The progress of different GANs on the synthetic ring example. . 85
4.1.2 The progress of different GANs on the synthetic grid example. . 86
4.1.3 Generated samples from a trained BuresGAN, with a DCGAN architecture. 87
4.1.4 Images generated by BuresGAN with a ResNet architecture. . . 90
4.2.1 Probability Density Function and RLS of a sample of this PDF. 93
4.2.2 RLS distribution using a Gaussian feature map on the unbalanced Ring and Grid data. 96
4.2.3 Visualization of the generation quality on Ring and Grid. . . . 100
4.2.4 Number of training samples in each mode for the Ring dataset. . 101
4.2.5 Generated images from unbalanced 012-MNIST. 102
4.2.6 Generated images from 06-CIFAR10 by a StyleGAN2 + Aug. and by RLS StyleGAN2 + Aug. 105

5.1.1 Interface of the Matlab tool for exploring the latent space. . . . 113
5.1.2 Latent space of the MNIST digits dataset for the digits 0 and 1. 114
5.1.3 Exploration of the latent space of the MNIST digits data set. . 114
5.1.4 Exploring different regions of the latent space of the Yale Face Database. 115
5.1.5 Latent space of the Yale Facebase database B. 115
5.1.6 Exploring the space between the regions of the latent space in figure 5.1.4e. 116
5.1.7 Three examples of interpolation between two subjects of the Yale Face Database B along the fourth component. 117
5.1.8 The latent space for 785 ECG beat signals of the MIT-BIH Arrhythmia dataset projected on different principal components. 118
5.1.9 Exploring the first three principal components of the latent feature space for the MIT-BIH arrhythmia database for normal and paced beats. 119
5.1.10 The latent space for 1000 zeros of the MNIST digits data set. . 120
5.1.11 Exploration of the latent space in Figure 5.1.10. 121
5.2.1 Gen-RKM schematic representation. 125

5.2.2 MNIST: Scatter plot of latent variable distribution when trained on 10000 images ($s = 2$). 126
5.2.3 Generated samples from the model using CNN as explicit feature map in the kernel function. 131
5.2.4 Learned latent space visualization of the Sketchy dataset. 132
5.2.5 Multi-view generation on Sketchy dataset showing labels, images and sketches generated. 134
5.2.6 Multi-view Generation (images and labels) on various datasets using implicit and explicit feature maps. 135
5.2.7 Exploring the learned uncorrelated features by traversing along the eigenvectors. 136
5.3.1 Schematic illustration of St-RKM training problem. 138
5.3.2 Traversals along principal components. 146
5.3.3 Samples of randomly generated batch of images. 150

6.1.1 Illustration of robustness against outliers on the MNIST dataset. 152
6.1.2 Illustration of robust generation on the MNIST dataset. 154
6.1.3 Illustration of robust denoising on the MNIST dataset. 156
6.1.4 Illustration of disentanglement on the 3DShapes dataset. 158
6.1.5 Illustration of latent traversals along the 3 latent dimensions for 3DShapes dataset using the robust Gen-RKM model. 159
6.2.1 Illustration of kernel MRCD on two toy datasets. 163
6.2.2 Results of the non-kernel MCD method on the toy datasets. . . 177
6.2.3 Kernel MRCD results on the toy datasets of Figure 6.2.1. 178
6.2.4 Illustration of the non-elliptical simulation setting with data generated from the t copula. 183
6.2.5 llustration of the non-elliptical simulation setting with data generated from the circle manifold. 184
6.2.6 Food industry example with KMRCD. 186
6.2.7 MNIST denoising results. 188
6.2.8 The mean absolute error of the denoised images to the original test images in function of the number of eigenvectors. 189

A.3.1 Illustration of sampling methods on an artificial clustering problem. 201
A.3.2 Clustering results. 202
A.3.3 Kernel approximation results. 203
A.3.4 Results accompanying the large-scale Nyström and KPCA experiments. 203
A.3.5 KPCA results. 204
A.3.6 Regression results. The condition number and MAPE on the test set are plotted as a function of $\log(\det(K_{\mathcal{CC}}))$. 204

A.3.7 Results accompanying the large-scale regression experiments. The condition number as a function of $\log(\det(K_{CC}))$ and the approximate RLS distribution are visualized. 204
A.3.8 Results accompanying the large-scale Nyström experiments. The condition number of K_{CC} versus the sample diversity. 205

B.5.1 The Relative operator norm of the Nyström approximation error as a function of the bandwidth σ. 222
B.5.2 The relative max norm of the Nyström approximation error as a function of the bandwidth σ. 222
B.6.1 Singular value spectrum of the data sets of Table 3.1.1 on a logarithmic scale. 222
B.6.2 The approximate ridge leverage scores accompanying Figure 3.1.6. 223
B.6.3 Diversity, relative max norm, and computation time as a function of the number of landmarks. 224

C.3.1 Generated images for the Stacked MNIST dataset. 232
C.3.2 Generated images for CIFAR-10 using a DCGAN architecture. . 233
C.3.3 Generated images for CIFAR-100 using a DCGAN architecture. 234
C.3.4 Generated images for STL-10 using a DCGAN architecture. . . 235

D.2.1 Number of modes and sample quality on the Ring dataset. . . . 245
D.4.1 Samples per mode for a single GAN trained on the Ring dataset. 249
D.4.2 Samples per mode for a single GAN trained on the Grid dataset. 249

E.4.1 Relative importance matrix as computed by Lasso and Random Forest regressors on DSprites dataset. 259
E.4.2 Relative importance matrix as computed by Lasso and Random Forest regressors on Teapot dataset. 260
E.4.3 Visualization of the toy dataset together with the first 5 Principal Components (PCs) of the latent space of the Gen-RKM model. . 261
E.4.4 Visualization of the traversals along the Principal components. 262
E.4.5 Comparing Gen-RKM and standard VAE for reconstruction and generation quality. 263

F.3.1 Samples of randomly generated images. 270
F.3.2 Loss evolution (log plot) during the training. 271
F.4.1 Reconstruction errors (log-scale) of various models during the training process over 1000 epochs. 272
F.4.2 Scatter plots of latent variables with histograms. 273
F.5.1 β-VAE and Info-GAN: Traversals along principal components. . 274

G.4.1 Illustration of the non-elliptical simulation setting. 284

List of Tables

2.2.1 Datasets and parameters used in the experiments. 44

3.1.1 Small scale data sets and parameters 63
3.1.2 Medium scale data sets and parameters 66

4.1.1 Experiments on the synthetic data sets. 87
4.1.2 KL-divergence between the generated distribution and true distribution for an architecture with 3 conv. layers for the Stacked MNIST dataset. 88
4.1.3 Stacked MNIST experiment for an architecture with 4 conv. layers. 88
4.1.4 Generation quality on CIFAR-10, CIFAR-100, and STL-10 with DCGAN architecture. 90
4.1.5 Best achieved IS and FID, using a ResNet architecture. 91
4.2.1 Experiments on the synthetic datasets Ring and Grid. 100
4.2.2 Experiments on the unbalanced 012-MNIST dataset. 103
4.2.3 Experiments on the unbalanced MNIST dataset. 104
4.2.4 Experiments on the unbalanced 06-CIFAR10 and unbalanced 016-CIFAR10 dataset. 105

5.2.1 FID Scores. 133
5.2.2 Disentanglement Metric on DSprites and Teapot dataset with Lasso and Random Forest regressor. 137
5.3.1 FID Scores for 8000 randomly generated samples. 142
5.3.2 Training times. 146
5.3.3 Sliced Wasserstein Distance of randomly generated 8000 samples. 148
5.3.4 Eastwood framework's disentanglement metric with Lasso and Random Forest regressor. 148

6.1.1 FID Scores for randomly generated samples when the training data is contaminated. 157
6.1.2 Disentanglement Metric on DSprites and 3D Shapes dataset. . . 160

6.1.3 Model architectures. 161
6.2.1 Kullback-Leibler deviations of $\hat{\Sigma}$ from Σ. 181
6.2.2 Speedup factors of KMRCD relative to MRCD. 182
6.2.3 Average number of outliers in the h-subset H, and among the $n(1-\varepsilon)$ points with lowest (kernel) Mahalanobis distance. . . 184

A.2.1 Datasets and parameters used for the experiments on the Nyström approximation. 199
A.3.1 Datasets and parameters used in the clustering experiments. . . 200
A.3.2 Datasets and parameters used for the experiments. 202

B.6.1 Parameters for the samplings with BLESS. 223

C.2.1 Average time per iteration in seconds for the convolutional architecture. 228
C.2.2 Inception Score for the best trained models on CIFAR-10, CIFAR-100 and STL-10, with a DCGAN architecture. 228
C.2.3 Comparison for Frobenius norm. Experiments on the synthetic data sets. 229
C.2.4 KL-divergence between the generated distribution and true distribution. 230
C.2.5 Ablation study on real image data sets for a convolutional architecture. 230
C.2.6 Ablation study for Stacked MNIST experiment for an architecture with 4 convolutional layers. 231
C.4.1 The generator and discriminator architectures for the synthetic examples. 236
C.4.2 The MDGAN encoder model and VEEGAN stochastic inverse generator architectures for the synthetic examples. 236
C.4.3 The hyperparameters and architectures of the stacked MNIST experiment performed by the different methods. 237
C.4.4 The generator and discriminator architectures for the Stacked MNIST experiments with 4 layers for both generator and discriminator. 237
C.4.5 The generator and discriminator architectures for the Stacked MNIST experiments. 238
C.4.6 Model architecture for the Stacked MNIST experiments. 238
C.4.7 The generator and discriminator architectures for the CIFAR-10 and CIFAR-100 experiments. 239
C.4.8 The MDGAN encoder model architecture for the CIFAR-10 and CIFAR-100 experiments. 239
C.4.9 The generator and discriminator architectures for the STL-10 experiments. 240

C.4.10 The MDGAN encoder model architecture for the STL-10 experiments. 240
C.4.11 The generator and discriminator ResNet architectures for the CIFAR-10 experiments. 241

D.1.1 Experiments on the UNBALANCED MNIST dataset. 244
D.2.1 Ablation study over the parameter γ on the synthetic datasets. 245
D.2.2 Ablation study over the regularization parameter γ and dimension k on the UNBALANCED 012-MNIST dataset. 246
D.2.3 Ablation study over the regularization parameter γ and dimension k on the UNBALANCED MNIST dataset. 247
D.3.1 Total training time in seconds, averaged over 10 runs. 248
D.5.1 The generator (left) and discriminator (right) architectures for the synthetic examples. 251
D.5.2 The generator and discriminator architectures for the UNBALANCED 012-MNIST and the UNBALANCED MNIST experiments. 252
D.5.3 The CNN architecture of the classifier used during the evaluation of the MNIST experiments. 252

E.3.1 Datasets and hyperparameters used for the experiments. 256
E.3.2 Details of model architectures used in Section 5.2. 257
E.4.1 Training time per epoch comparisons on MNIST and CelebA datasets. 260

F.2.1 Datasets and hyperparameters used for the experiments. 268
F.2.2 Model architectures. 268
F.3.1 Training timings per epoch (in minutes) and disentanglement scores for different variants of RKM when trained on the 3Dshapes dataset. 269
F.3.2 FID scores computed on randomly generated 8000 images. . . . 269

G.3.1 The KL divergence in function of the condition number of the ALYZ covariance matrix. 279
G.3.2 Averaged computation times of the four initial estimators. . . . 280
G.3.3 The number of times each initial estimator provided the best solution (i.e. the lowest covariance determinant) after C-steps. . . 281
G.3.4 MSE for Σ of type ALYZ (multiplied by 1000). 282
G.3.5 Regularization coefficients for Σ of type ALYZ. 283

Chapter 1

Introduction

Artificial intelligence and machine learning are becoming more and more part of our everyday lives. For example, the recommendation engine of Netflix recommends over 75% of what we watch. Facebook uses automatic tagging of friends and families based on facial features. Alexa of Amazon is a virtual assistant based on natural language processing and machine learning that can update your grocery list, play songs, connect with other devices in your home,... Additionally, machine learning has proved to have superhuman abilities in numerous fields. An example is AlphaGo, the first computer program to defeat Lee Sedol, a Go world champion and one of the strongest Go players in history [2]. In 2020, Deepmind built an AI system for breast cancer screening that out-performed a test group of six radiologists [1]. AI systems also had tremendous success in games. Examples are Dota 2 [172], StarCraft II [3], or chess [214].

Machine learning can roughly be divided into supervised learning, which trains algorithms based on input-output pairs that are labeled by humans, and unsupervised learning that uses unlabeled data to extract structure from the data. Typical examples of supervised learning are classification and regression. For example, a user wants to filter out spam emails or use historical stock market information to anticipate upcoming fluctuations. Examples of unsupervised learning are outlier detection for fraudulent credit card purchases or generative models that create new art designs.

1.1 With great power comes great responsibility

As machine learning becomes more integrated into our everyday lives, people are becoming more interested in the *fairness*, *interpretability*, and *robustness* of these systems. This made the AI research community emphasize more the role of trustworthy machine learning. In [213], three levels of organizational structures for trustworthy AI are described: (1) *Reliable systems based on sound software engineering practices.* Machine learning systems should be reliable so that they work consistently on every kind of circumstances they can encounter. Examples of improving reliability are audit trails to analyze failures, software engineering workflows like cleaning, labeling, and data analytics to understand abnormal distributions, errors, and anomalies. Other aspects are bias mitigation to enhance fairness, explainable user interfaces, and verification. Verification techniques verify if a certain property holds. Typical verification questions are robustness against adversarial attacks [89] or fairness [71]. Once it is verified that a certain property is violated, the user can take action like e.g. adding the adversarial example to the training set. (2) *Safety culture through business management strategies.* The management should invest in a safety culture within the organization, which can be achieved by hiring and training specifically oriented to safety, reporting of failures, internal review boards for problems and future plans, and the alignment with industry-standard practices. (3) *Trustworthy certification by external reviews.* These independent reviews would boost public wide acceptance for AI. The certification comes from the government, accounting firms conducting external audits, insurance companies compensating for failures, nongovernmental and civil society organizations, or research institutes. A similar distinction is made in [229]. The authors discuss four categories of trustworthiness technologies for machine learning, namely fairness, explainability, auditability, and safety. It is stressed that the implementation of such technological solutions in itself will not make the system trusted, but they are used to improve the trustworthiness of the system.

Other aspects of machine learning that are closely related to trustworthiness are responsible AI, ethical AI, privacy, and security. An overview of opportunities, risks, and recommendations of ethics and AI can be found in [77]. The authors present a number of ethical principles together with recommendations that should support the development and adoption of AI. Worries about privacy and security have led to the development of the General Data Protection Regulation (GDPR) in the EU [5]. The GDPR's primary aim is to enhance the control and rights of individuals over their personal data. This need for privacy and security is also important in machine learning itself. An overview of privacy-preserving machine learning algorithms can be found in [114]. As the name suggests, the objective is to train machine learning algorithms without revealing the identity

of individuals. In ethical AI [36], AI systems should do what is right, fair, and just without causing harm. Some issues that are tackled are liability for self-driving cars, weaponization of AI, the ability of robots to harm people, etc.

In this thesis, multiple algorithms are described that machine learning practitioners could use to build more reliable systems, hence improving trustworthiness. These algorithms can be divided into three different use-cases:

1. Fairness and bias in machine learning
2. Interpretability in generative modeling.
3. Robustness against outliers and noise

Fairness and bias in machine learning. Bias in the data can occur in a number of ways, an example is population or sampling bias. Data is often partitioned into different subpopulations where the frequency of each subpopulation can be different. Think for example of a medical dataset that contains data from different diseases, that occur with different frequencies. When training machine learning models on these datasets, it is important to not only predict well in the majority of the data but also for specific outlying points, as incorrectly classifying these rare points could lead to significant harm to patients. Solutions to bias in machine learning can roughly be divided into three categories: pre-processing, in-processing, and post-processing [37]. Post-processing approaches apply transformations to the model output to improve prediction fairness [94]. First, an unfair classifier is trained, afterwards, the outputs of the classifier are adjusted to enforce fairness. Although this approach is likely to decrease the bias of the classifier, separating the training from the fairness enforcement could lead to a sub-optimal trade-off between fairness and accuracy [246]. In-processing approaches incorporate one or more fairness metrics into the model optimization function [255, 7]. A potential downside of this approach is that this extra complexity could lead to more difficult or unstable training. The above approaches address the problem of bias by adjusting the machine learning model. However, it is oftentimes the training data itself that is biased, and thus it is potentially more effective to tackle this problem directly. This brings us to the final approach. Pre-processing approaches tend to alter the sample distributions to remove bias from the data. Afterwards, the model is trained on the 'repaired' data. An example is given in the context of risk-averse learning, the authors of [51] discuss an adaptive sampling such that the model not only performs well on average but also on rare data points. We speak of hidden stratification in machine learning when models do not sufficiently take into account 'hidden' small populations in the training dataset, i.e. the small subpopulations are unlabeled or belong to a

bigger more general class. For example, a population labeled car could contain multiple subpopulations composed of red cars, green cars, jeeps, trucks, etc. As these subpopulations are hidden from us, there is the need for an unsupervised method to sample from each subpopulation. In this thesis, we analyze diversity sampling to reduce bias and improve regularization. An example of the dangers of bias in data is given in Figure 1.1.1. An image upscaling application using a StyleGAN is trained on a dataset using faces [4]. The goal is to generate a high pixel image, given a low-quality one. As can be seen from the example, the application is producing white faces much more frequently than faces of people of color. This bias is likely inherited from the dataset, as the dataset was skewed toward a single demographic.

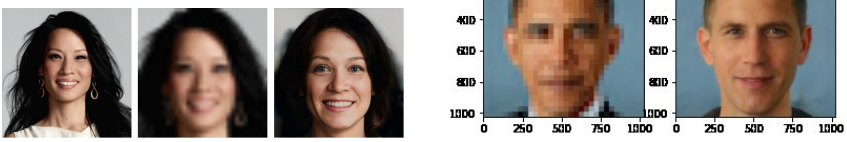


Figure 1.1.1: Image is taken from [4]. The first subfigure is the low-quality input (real image), the second subfigure is the original image (real image), the third subfigure is the upscaled generated image by the StyleGAN (fake image). The fourth subfigure contains a downsampled image of Obama (real image), the last figure is the upscaled image generated by the application (fake image).

Interpretability in generative modeling. A second focus of this thesis is interpretability [142]. For a user to gain trust in the model, it is beneficial to provide some interpretable information. For example, medical experts want to understand the decision behind a model diagnosis. Or in credit assessments models, where credit denial has to be legally supported by a clear motivation. Models like linear regression and logistic regression are popular choices for these types of problems, as the models are inherently interpretable due to their linear nature. Unfortunately, for more advanced problems, these models are deemed too simple. Therefore, several works have focused on explaining deep neural network models. Examples are surrogate models like LIME [189] or SHAP [150]. These methods approximate locally the black-box model with a surrogate model to explain the decision in the neighborhood of a data point. Other approaches have focused on visualization techniques of neural networks [259]. An overview of interpretability in machine learning can be found in [86, 20]. The above approaches are examples of interpretability in supervised learning, this thesis focuses on interpretability in generative modeling. In representation learning and generative modeling, researchers have often turned to *disentanglement* [99],

which measures how well individual components in the latent space match ground-truth factors that generated the data. An example is a dataset of chair designs. The ground-truth generating factors are e.g. color, shape, size, type,... In a disentangled representation, each of the components would match one of the aforementioned factors. When generating a new chair design, the user could use this interpretable information to carefully guide the design process to the desired outcome, e.g. a white chair of predefined height where the rest of the features are determined by the model. This process is also called pseudo-generation.

Robustness against outliers and noise. Outliers are points that defer from the majority of the regular observations. These may be caused by human error, a change in system behavior, or faulty sensors. Outliers have a harmful effect on the machine learning model and need to be detected. When these outliers are correctly identified, further investigation can focus on the underlying cause. In supervised anomaly detection, the data has to be labeled as normal or abnormal and involves training a classifier. In unsupervised anomaly detection, the data is unlabeled. The goal is to find a fit that is close to the fit we would have found without the outliers. Outliers are then flagged by their large deviation from that robust fit. To determine a robust fit, techniques often assume that the majority of the instances in the dataset are normal, outliers correspond to instances that least match the remainder of the dataset. Some examples of unsupervised outlier detection techniques are one-class SVMs [206], classical robust statistics approaches [195], clustering-based methods [97], or density-based techniques [27]. The robustness of estimators is typically measured by their breakdown value, which is the smallest proportion of observations in the dataset that need to be replaced to carry the estimate arbitrarily far away. For example, the mean has a breakdown value of 0%, a single outlier can make the mean arbitrarily large. A robust alternative is the median estimator, which has a breakdown value of about 50%. The minimum covariance determinant estimator (MCD) [108] is a highly robust estimator of location and scatter, with a maximal breakdown value of 50% when half of the points are chosen as regular observations (see Section 1.2.2 hereafter). Due to the high robustness, the MCD estimator plays a central role in this thesis. A different notion of robustness is adversarial robustness, where users want to test the worst-case robustness of machine learning algorithms. This is done by adversarial attacks or examples. Adversarial examples can be defined as inputs specifically designed to force a machine learning system to produce large errors. Adversarial robustness has been studied in supervised learning [35, 222] and unsupervised learning [34].

1.2 Background information

1.2.1 The kernel trick

A popular class of machine learning models is kernel methods. These methods became very popular in the 1990s by the work of Cortes and Vapnik [49], which extended the linear support vector machine (SVM) to deal with nonlinearities by introducing a nonlinear mapping $\phi(\cdot)$. By first mapping the data to a higher-dimensional space, it is possible for the method to find linear patterns which would correspond to nonlinear patterns in the original space. An example of this can be found in Figure 1.2.1. Instead of calculating this feature map directly, one can implicitly work with it through the use of a positive definite kernel function.

Definition 1.2.1. *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel on $\mathcal{X}$ if there exists a real Hilbert space $\mathcal{F}$ and a map $\phi : \mathcal{X} \to \mathcal{F}$ such that for all $\boldsymbol{x}$, $\boldsymbol{y}$ in $\mathcal{X}$:*

$$k(\boldsymbol{x}, \boldsymbol{y}) = (\phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y})),$$

where ϕ is called a feature map and $\mathcal{F}$ is called a feature space.

Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be a dataset of points in $\mathbb{R}^d$, the kernel matrix is given by $K = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij=1}^n$ and defines the similarity between data points. The use of kernels makes it possible to operate in a high-dimensional, implicit feature space without computing the coordinates of the data in that space, but rather by replacing inner products with kernel matrix entries. By making use of the kernel trick, many different linear methods have been kernelized. We will now discuss some examples in next the part.

Kernel principal component analysis

A well-known example of the kernel trick is kernel principal component analysis (PCA) [208], where linear PCA is performed in a kernel-induced feature space instead of the original space. Linear PCA is a basis transformation to diagonalize the empirical covariance matrix of the data $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i\in[n]}$:

$$\Sigma = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top,$$

where we assume the data to be centered, i.e. $\sum_{i=1}^n \boldsymbol{x}_i = 0$. PCA is commonly used for dimensionality reduction by projecting each data point onto the first

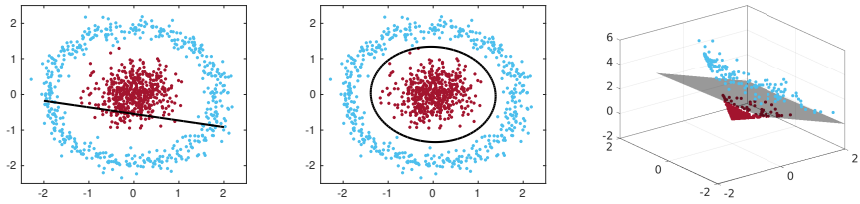


Figure 1.2.1: Illustration of the kernel trick on a binary classification problem. The dataset is an enclosed circle, samples from the two classes are depicted in red and blue. The classification boundary of the trained SVM classifier is shown in black. The figure on the left shows the classification of a linear SVM. The figure in the middle is a non-linear SVM trained with a second-degree polynomial kernel. The figure on the rights shows a linear SVM trained on the transformed data using the explicit feature map of the polynomial kernel.

few principal components while preserving as much of the data's variation as possible. This works very well when the data is sampled from an elliptical distribution. Unfortunately, this is not always the case in practice. Kernel PCA is the generalization of linear PCA to a kernel-induced feature space $\mathcal{F}$, where one makes use of the mapping $\phi(\cdot)$ [207]. The covariance matrix now becomes

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{x}_i)\phi(\boldsymbol{x}_i)^\top,$$

where we again assume the data to be centered, i.e. $\sum_{i=1}^{n}\phi(\boldsymbol{x}_i) = 0$. We determine the eigenvalues $\lambda \geq 0$ and eigenvectors $V \in \mathcal{F}\backslash\{0\}$ satisfying $\lambda V = \Sigma V$. All solutions of V lie in the span of $\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)$, we can thus consider the equivalent system

$$\lambda\left(\phi(\boldsymbol{x}_k)\cdot V\right) = \left(\phi(\boldsymbol{x}_k)\cdot \Sigma V\right) \text{ for all } k = 1, \ldots, n.$$

There exist coefficients $\alpha_1, \ldots, \alpha_n$ such that $V = \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i)$. Combining the above equations, we get the following eigenvalue problem: $n\lambda\boldsymbol{\alpha} = K\boldsymbol{\alpha}$. The projection of a point $\boldsymbol{x}$ on the k-th component is given by:

$$\left(V^k\cdot\phi(\boldsymbol{x})\right) = \sum_{i=1}^{n}\frac{\alpha_i^k}{\sqrt{\lambda^k}}\left(\phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x})\right) = \sum_{i=1}^{n}\frac{\alpha_i^k}{\sqrt{\lambda^k}}K(\boldsymbol{x}_i,\boldsymbol{x}).$$

The least-squares support vector machine (LS-SVM) formulation of kernel PCA can be found in [223], which views PCA as a one-class modeling problem with zero target value around which one maximizes the variance. Applications of kernel PCA are non-linear feature extraction [207], outlier detection [102], or denoising [207].

Restricted kernel machines. Recently, kernel PCA has been cast in the restricted kernel machine (RKM) framework [224]. By making use of conjugate feature duality, kernel PCA has a representation with visible and hidden units. Given $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and centered feature-map $\phi(\cdot)$, the LS-SVM formulation of kernel PCA [225] is written as:

$$\min_{U,\boldsymbol{e}_i} \frac{1}{2}\operatorname{Tr}(U^\top U) - \frac{1}{2\lambda}\sum_{i=1}^N \boldsymbol{e}_i^\top \boldsymbol{e}_i$$

$$\text{s.t. } \boldsymbol{e}_i = U^\top \phi(\boldsymbol{x}_i) \quad \forall i = 1,\ldots,N,$$

with $U \in \mathbb{R}^{d\times s}$ the interconnection matrix. From the KKT conditions we obtain the following eigenvalue problem: $K\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$. By using the notion of conjugate feature duality, introduced in [224], the error variables $\boldsymbol{e}_i$ are conjugated to latent variables $\boldsymbol{h}_i$ using:

$$\frac{1}{2\lambda}\boldsymbol{e}^\top\boldsymbol{e} + \frac{\lambda}{2}\boldsymbol{h}^\top\boldsymbol{h} \geq \boldsymbol{e}^\top\boldsymbol{h}, \qquad \forall \boldsymbol{e},\boldsymbol{h} \in \mathbb{R}^s$$

This is also known as the Fenchel-Young inequality for quadratic functions [190]. By eliminating the variables $\boldsymbol{e}_i$ and using the above equation, we obtain the RKM training objective function:

$$\begin{aligned} J &= \frac{1}{2}\operatorname{Tr}\left(U^\top U\right) - \frac{1}{2\lambda}\sum_{i=1}^N \boldsymbol{e}_i^\top \boldsymbol{e}_i \quad \text{s.t. } \boldsymbol{e}_i = U^\top\varphi\left(\boldsymbol{x}_i\right), \forall i \\ &\leq -\sum_{i=1}^N \boldsymbol{e}_i^\top \boldsymbol{h}_i + \frac{\lambda}{2}\sum_{i=1}^N \boldsymbol{h}_i^\top\boldsymbol{h}_i + \frac{1}{2}\operatorname{Tr}\left(U^\top U\right) \quad \text{s.t. } \boldsymbol{e}_i = U^\top\varphi\left(\boldsymbol{x}_i\right), \forall i \quad (1.2.1) \\ &= \sum_{i=1}^N \left(-\phi(\boldsymbol{x}_i)^\top U\boldsymbol{h}_i + \frac{\lambda}{2}\boldsymbol{h}_i^\top\boldsymbol{h}_i\right) + \frac{1}{2}\operatorname{Tr}(U^\top U). \end{aligned}$$

The objective resembles the energy form of restricted Boltzmann machines (RMBs) [100], which are generative artificial neural networks. This motivates the search towards a generative model using the RKM framework, proposed in Chapter 5.

Kernel ridge regression

A well-known kernel method for supervised learning is kernel ridge regression (KRR). Given an Reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ with kernel K,

the objective for the input-output pairs $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i\in[n]}$ is given by

$$f^\star = \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2 + \gamma\|f\|^2_{\mathcal{F}}, \text{ with } \gamma > 0.$$

The first term is minimizing the error, and the second term acts as a regularizer. The regressor is $f^\star(\cdot) = \sum_{i=1}^{n}\alpha_i^\star k(\boldsymbol{x}_i, \cdot)$ with $\boldsymbol{\alpha}^\star = (K + n\gamma\mathbb{I})^{-1}y$. When a bias term is included, the model corresponds to an LS-SVM [225]. To scale KRR to large-scale datasets, multiple ensemble approaches have been proposed [261, 107, 140]. In these ensemble approaches, the data points are divided into smaller subsets and estimators are trained on the divisions.

Nyström Approximation.

Kernel methods have difficulty scaling to large-scale datasets, as each of these methods requires the construction of the $n \times n$ kernel matrix. Constructing the kernel matrix already requires n^2 operations. Solving a linear system involving K has a complexity $O(n^3)$. Therefore, improving the scalability of kernel methods has been an active research question. A commonly used method is the Nyström Approximation [244]. Given data $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i\in[n]}$ and an $n \times n$ kernel matrix $K = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$. Define a subset of landmarks $\mathcal{C} \subseteq [n]$ and a $n \times |\mathcal{C}|$ sampling matrix C obtained by selecting the columns of the identity matrix indexed by $\mathcal{C}$. This gives the submatrices: $K_{\mathcal{C}} = KC$ and $K_{\mathcal{C}\mathcal{C}} = C^\top KC$. Then, the $n \times n$ kernel matrix K is approximated by a low rank Nyström approximation:

$$\hat{K}(K, \mathcal{C}) = K_{\mathcal{C}}K_{\mathcal{C}\mathcal{C}}^{-1}K_{\mathcal{C}}^\top.$$

Solving a linear system with the Nyström approximation only has a computational complexity of $O(n|\mathcal{C}|^2)$. Notice that the inverse can be badly conditioned when the subset of landmarks is badly chosen. The Nyström approximation has been successfully used to scale-up kernel methods like kernel PCA [220], kernel ridge regression [200, 136, 201], LS-SVM [225], and kernel k-means [240]. Another low-rank approach for large-scale kernel methods is Random Fourier features [185].

1.2.2 The determinant of the kernel matrix

The determinant of the kernel matrix plays a central role in the thesis. This is the main tool that is used to measure the diversity of a dataset. Depending on the application, diversity can be desired: the dataset is stratified and contains multiple smaller subpopulations. Diversity is not desired when the user suspects

the data to contain outliers. We now discuss two popular methods that use the determinant to measure diversity.

Determinantal point processes. Determinantal point processes (DPPs) are an unsupervised diversity sampling technique. DPPs have been used in applications like improving diversity in recommender systems [40], diverse human motion forecasting [254], image search tasks [124], and many others. A point process $\mathcal{P}$ on a ground set $\mathcal{Y} = \{1, 2, ..., n\}$ is a probability measure over point patterns of $\mathcal{C}$, which are finite subsets of $\mathcal{Y}$. When $\mathcal{C}$ is a random subset, drawn according to the DPP $\mathcal{P}$, we have:

$$\mathcal{P}(\mathcal{C} \subseteq \mathcal{Y}) = \det(K_{\mathcal{C}\mathcal{C}}), \tag{1.2.2}$$

with the marginal kernel $K \preceq \mathbb{I}$ a positive symmetric semidefinite matrix with all eigenvalues smaller than or equal to 1, containing the elements of $\mathcal{Y}$. From above equation follows:

$$\begin{aligned}\mathcal{P}(i \in \mathcal{Y}) &= K_{ii} \\ \mathcal{P}(i, j \in \mathcal{Y}) &= K_{ii}K_{jj} - K_{ij}K_{ji} \\ &= \mathcal{P}(i \in \mathcal{Y})\mathcal{P}(j \in \mathcal{Y}) - K_{ij}^2.\end{aligned}$$

The entries of the marginal kernel are measurements of similarity between datapoints, hence highly similar elements are unlikely to appear together. More precisely, the diagonal elements of the kernel matrix give the marginal probability of inclusion for individual elements of $\mathcal{Y}$, whereas the off-diagonal elements determine the repulsion between pairs of elements. Thus for large values of K_{ij}, or a high similarity, points are unlikely to appear together. For example, if $K_{ij} = \sqrt{K_{ii}K_{jj}}$, the two point are perfectly similar and do not appear together almost surely. Contrarily, when there are no similarities or correlations in the data, K is diagonal and all elements appear independently. DPPs also have a intuitive geometric interpretation. If we write $K = B^\top B$ with B a matrix of size $b \times n$. The probability of sampling a subset is $\mathcal{P}(\mathcal{C} \subseteq \mathcal{Y}) \propto \mathrm{Vol}^2\left(\{B_i\}_{i \in \mathcal{C}}\right)$, where the right hand side is the squared $|\mathcal{C}|$-dimensional volume of the parallelepiped spanned by the columns of B corresponding to elements in $\mathcal{C}$. Thus, diverse sets are more probable because their feature vectors are more orthogonal, and hence span larger volumes. Items with parallel feature vectors are sampled together with probability zero, since their feature vectors define a degenerate parallelepiped.

An example of diversity sampling using a k-DPP is given in Figure 1.2.2. A k-DPP is a conditional DPP that models only sets of cardinality k. The exact sampling of a DPP or a k-DPP from a set of n items takes $O(n^3)$ operations. Therefore, several improvements or approximations of the sampling algorithm

have been proposed in [58], [136], [135] and [232]. There also exist other methods to sample diverse landmarks, such as volume sampling [59], greedy methods [46], ridge leverage score (RLS) sampling [9], etc. RLSs are a measure of the uniqueness of data points and are defined as follows:

$$\ell_{n\gamma,K}(\boldsymbol{x}_i) = [P_{n\gamma}(K)]_{ii},$$

with $i \in [n]$ and projector kernel matrix $P_{n\gamma}(K) = K(K + n\gamma\mathbb{I})^{-1}$ with K any PSD kernel matrix and regularization term γ. Points are sampled proportional to their RLS. Compared to DPP sampling, RLS sampling is an independent sampling technique: sampling a single point does not affect the sampling probability of the next point. Note that the subset is generally less diverse due to the independence of sampling than when using DPPs. RLS sampling is frequently used in matrix approximation due to its good theoretical guarantees [9]. There exist multiple approximations for large-scale data [200, 165].

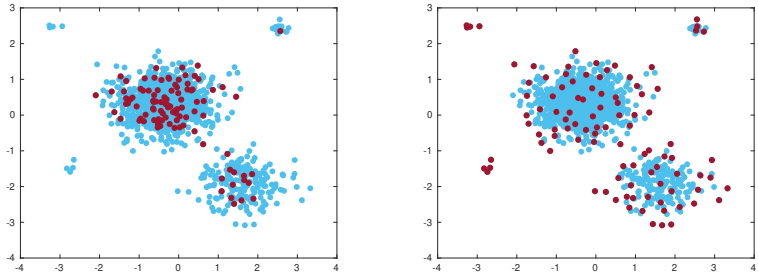


Figure 1.2.2: Illustration of k-DPP sampling on a dataset containing 5 different subpopulations sampled from different Gaussian distributions. Samples from the dataset are depicted in blue, sampled data points in red. The uniform sampling procedure (left) mostly selects points from the majority cluster, a small number of points from the core of the second majority cluster, whereas only 1 point from the 3 smaller clusters. Using a k-DPP sampling procedure (right) results in an even spread over the full dataset.

Minimum covariance determinant. The minimum covariance determinant (MCD) estimator is a highly robust estimator of multivariate location and scatter [192]. The objective is to find the h observations out of n whose covariance matrix has the lowest determinant. Assume that we have a d-variate dataset X containing n data points. We do not know in advance which of these points are outliers, and they can be located anywhere. The MCD aims to minimize the determinant of $\hat{\Sigma}^H$ among all $H \in \mathcal{H}$, where the latter denotes the collection of all possible sets H with $|H| = h$:

$$\hat{\Sigma}_{\text{MCD}} = \underset{H \in \mathcal{H}}{\text{argmin}} \ \det(\hat{\Sigma}^H) \ ,$$

with $\hat{\Sigma}^H = \frac{1}{h-1}\sum_{i\in H}(\boldsymbol{x}_i - \boldsymbol{c}^H)(\boldsymbol{x}_i - \boldsymbol{c}^H)^\top$ and $\boldsymbol{c}^H = \frac{1}{h}\sum_{i\in H}\boldsymbol{x}_i$. An example of the MCD estimator is given in Figure 1.2.3. Notice that in this scenario, diversity is not desired. The determinant should be as small as possible to detect and omit potential outliers. Computing the exact MCD has combinatorial complexity, which makes the computation only feasible for tiny datasets. The approximate FastMCD algorithm was proposed to alleviate this problem [199] using nesting and subsampling techniques. Afterwards, the DetMCD algorithm was proposed [112]. It is fully deterministic, more robust than FastMCD, and needs less computation time. The DetMCD algorithm uses an ensemble of six initial estimators to ensure high robustness against different contamination patterns. Although these estimators make it possible to estimate the MCD on larger datasets, high-dimensional datasets remain a problem. The MCD approach requires that the dimension d must satisfy $d < h$ for the covariance matrix of any h-subset to be non-singular, where it is often recommended to take $n > 5d$ for higher accuracy. The minimum regularized covariance determinant estimator (MRCD) minimizes the determinant of a convex combination of a target matrix and the sample covariance matrix of the subset [24]. This regularization mechanism ensures that the MRCD estimator is defined in any dimension, is well-conditioned by construction, and preserves the good robustness properties of the MCD. An overview of different MCD variants is found in [108]. Due to their high robustness, MCD estimators and variants were used to robustify several methods such as linear regression [198], PCA [111], and discriminant analysis [113]. However, two issues could arise when using these MCD-type estimators: 1) the estimators assume that the non-outlying observations are roughly elliptically distributed. 2) the computation time increases substantially when the number of variables goes up. Both issues are resolved by the introduction of the kernel minimum regularized covariance determinant estimator in this thesis (see Chapter 6).

Note that diversity sampling algorithms like DPP sampling have a high chance of sampling outliers (points that deviate from the majority of the data) as they would increase the determinant significantly. When we use diversity sampling, we thus assume that these 'outliers' or smaller subpopulations are important points in the data. Vice versa, outlier detection methods like MCD will always remove smaller subpopulations from the data, we here assume these points are unwanted or noise. The different assumptions may not always be correct, therefore, it is important to do further investigation on the underlying cause of why these points deviate from the majority of the data.

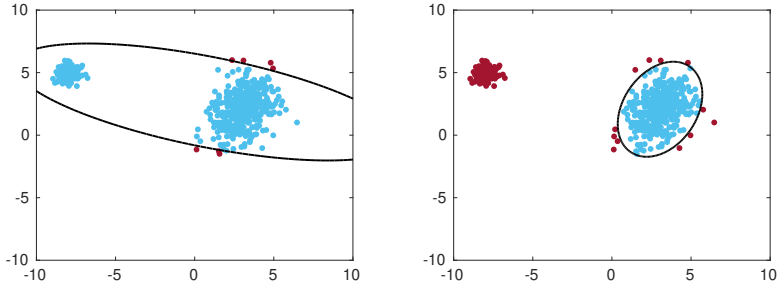


Figure 1.2.3: Illustration of the MCD estimator on an elliptical dataset. The dataset contains 20% of outlying observations. The generated regular observations are sampled from a normal distribution. Flagged outliers are marked in red, regular observation in blue. In the figure on the left, the classical covariance estimator is used and in the figure on the right, the robust MCD estimator with $h = \lfloor 0.75n \rfloor$ is used. The contours display the 97.5% tolerance ellipse. Only the MCD estimator is not affected by the noise.

1.2.3 Generative modeling

With the rise of deep learning, a new class of deep generative models has emerged. These models can learn any kind of data distribution in an unsupervised manner and have achieved tremendous success in just a few years. Two of the most popular methods are generative adversarial networks (GANs) and variational autoencoders (VAEs), which we will briefly discuss.

GANs. A GAN [88] consists of a discriminator $D : \mathbb{R}^d \to \mathbb{R}$ and a generator $G : \mathbb{R}^\ell \to \mathbb{R}^d$ which are typically defined by neural networks, and parametrized by real vectors. The value $D(\boldsymbol{x})$ gives the probability that $\boldsymbol{x}$ comes from the empirical distribution, while the generator G maps a point $\boldsymbol{z}$ in the latent space $\mathbb{R}^\ell$ to a point in the data space $\mathbb{R}^d$. The training of a GAN consists of solving

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))],$$

by alternating two phases of training. The first expectation is over the empirical data distribution p_d and the second is over the generated data distribution p_g, implicitly given by the mapping by G of the latent prior distribution $\mathcal{N}(0, \mathbb{I}_\ell)$. A schematic representation of a GAN is given in Figure 1.2.4. State-of-the-art GANs [118, 28] can produce good-looking high-resolution images that are often indistinguishable from real ones. However, getting consistent results over the full support of the dataset remains a difficult task. This problem is also known as mode collapse or mode dropping, where the trained model is incapable of

generating samples from all the modes in the data. A mode is defined as a local maximum in the distribution. To solve this problem, multiple GAN variants have been proposed [159, 217, 74, 83, 141, 62, 250]. The most notable are GDPP-GAN [74], which matches the eigenvectors and eigenvalues of the real and fake diversity kernel, and PacGAN [141], which proposes a simple packing procedure that significantly reduces mode collapse. Closely related is the problem of complete mode coverage, where trained models should be able to sample from all modes even when the number of training samples of a mode is very small. This is a common phenomenon in biased data. A solution to reduce selection bias in training data was proposed by [63], named importance weighted generative networks. Another approach to complete mode coverage by [263] iteratively trains a mixture of generators. At each iteration, the sampling probability is updated so that the probability to sample a missing mode is increased.

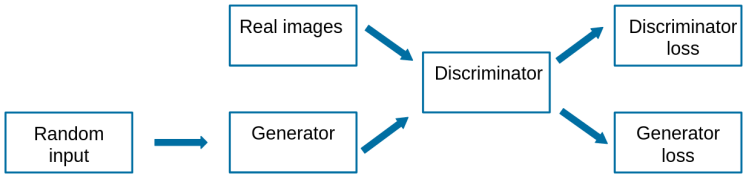


Figure 1.2.4: The objective of the generator is to "fool" the discriminator network by producing novel candidates that the discriminator thinks are part of the true data distribution. The discriminator is trained to distinguish between real and fake samples.

VAEs. Let $p(\boldsymbol{x})$ be the distribution of the data $\boldsymbol{x} \in \mathbb{R}^d$ and consider latent vectors $\boldsymbol{z} \in \mathbb{R}^\ell$ with the prior distribution $p(\boldsymbol{z})$, chosen to be a standard normal. Then, one defines an encoder $q(\boldsymbol{z}|\boldsymbol{x})$ given by $\mathcal{N}(\boldsymbol{z}|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), \gamma^2\mathbb{I})$, where the mean and variance are given by the neural network $\phi_{\boldsymbol{\theta}}$, parametrized by $\boldsymbol{\theta}$. A random decoder $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\psi_{\boldsymbol{\xi}}(\boldsymbol{z}), \sigma_0^2\mathbb{I})$ is associated to the decoder neural network $\psi_{\boldsymbol{\xi}}$, parametrized by $\boldsymbol{\xi}$, which maps latent codes to the data points. The VAEs are estimated by maximizing the lower bound

$$\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log(p(\boldsymbol{x}|\boldsymbol{z}))] - \beta\,\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}), p(\boldsymbol{z})) \leq \log p(\boldsymbol{x}),$$

which is called the Evidence Lower Bound (ELBO) when $\beta = 1$. The schematic representation of a VAE is given in Figure 1.2.5. VAEs and GANs have complementary properties. GANs can generate images of very high quality, but they tend to lack full support over the data and are difficult to train. On the

other hand, VAEs generate more blurry images but are better density models in terms of the likelihood criterion. Examples of artificial creativity using VAEs are the design of drug-like molecules and organic light-emitting diodes [87], or interpolating between sentences [25]. Some recent advances show promising results regarding high-quality generation using VAEs [235].

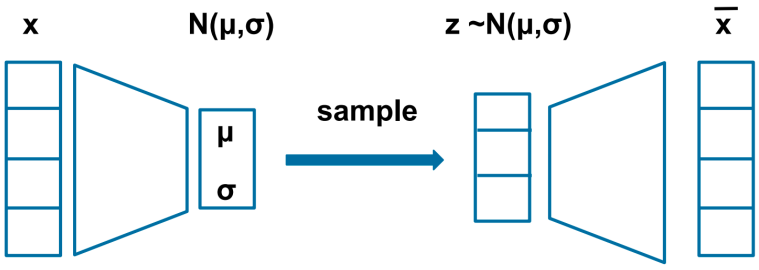


Figure 1.2.5: A VAE is an autoencoder whose encoding distribution is regularised during the training. This ensures that the latent space has good properties to generate new data. The data is mapped through the decoder network onto the latent space. Compared to a traditional AE, which encodes the input as a single point, the input is encoded as a distribution over the latent space. Afterwards, a point is sampled from the distribution and decoded by the decoder network.

Disentanglement. Generative methods like GANs and VAEs allow sampling high dimensional and difficult distributions like images, speech, text,... One of the challenges besides high-quality generation is *meaningful* generation. A solution is the use of latent space models with a disentangled representation [99]. An illustration of a latent traversal is shown in Figure 1.2.6, where one observes that only one specific feature of the image is changing as one moves along a component in latent space. For instance, in the first row of the right-hand side of Figure 1.2.6, we observe that moving along the first principal component generates images where only floor color is varying while all other features such as shape, scale, wall color, object color, etc. are constant. In the second row, the object scale is only changing. As previously mentioned, such a representation imparts more interpretability to the model. Several works have focused on analyzing disentanglement in VAEs [42, 31, 120]. In [191] is argued that the diagonal approximation in the encoder together with the inherent stochasticity forces the local orthogonality of the decoder. This local behavior of promoting both reconstruction and orthogonality matches closely how the PCA embedding is chosen. Experiments show that VAEs indeed strive for local orthogonality, and there is a clear correlation between orthogonality and the disentanglement score. This is a main motivation for the inclusion of a kernel PCA loss to the

VAE objective (see Chapter 5). A comprehensive study on disentanglement and many VAE variants is done in [146]. The role of disentanglement was also studied in GANs, where the InfoGAN [43] is one of the most known works. Other examples are [230, 239].

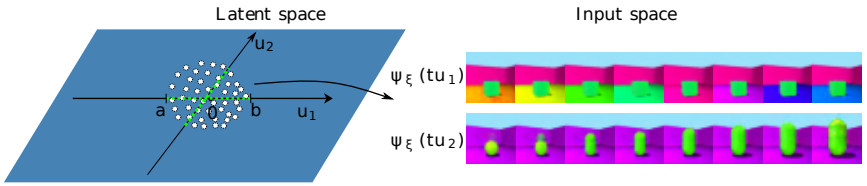


Figure 1.2.6: Image by the decoder of the latent space traversal, i.e., $\psi_{\boldsymbol{\xi}}(t\boldsymbol{u}_i)$ for $t \in [a, b]$ with $a < b$ and for some $i \in \{1, \ldots, m\}$. Yellow dots are training points encoded in the latent space. Green and black dashed-lines represent the walk along $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ respectively. At every step of the walk, the image of the decoder is computed to generate the data in the input space. The images were generated by St-RKM (see Chapter 5).

1.3 Research aims

We outline here a number of research aims, each clarified subsequently.

Aim: To analyze diversity sampling as a method to reduce bias in data.

Datasets often contain a number of small hidden subpopulations. To achieve close-to-optimal generalization performance, it is important that these small hidden subpopulations are not overlooked. In this thesis, we analyze diversity sampling as an unsupervised method to increases the probability of every subpopulation being included in the training process, hence reducing potential bias in the model. Second, we propose to split the evaluation metric into two parts: performance in the bulk and tail of data. By splitting the loss function into two parts, one can identify if the model is not only focusing on the majority of the data but also performing well in 'outlying' subpopulations. This aim is tackled in Chapter 2.

Aim: To provide new diversity sampling algorithms and establish a connection between current popular approaches.

Determinantal point processes [125] and leverage scores [73] are two of the most popular approaches for diversity sampling in kernel methods. Both these methods are connected by their use of the projector kernel matrix. Moreover, there is a known connection between Christoffel functions and ridge leverage scores [177]. In this thesis, we introduce conditioned Christoffel functions. These *conditioned* Christoffel functions are shown to be connected to *regularized* conditional probabilities for DPPs, which yields a relation between ridge leverage score sampling and DPP sampling. These new connections are used to introduce and analyze new diversity sampling algorithms in Section 3.1.

Aim: To analyze the role of diversity in GANs.

Mode collapse is a common problem in GANs [88], which happens when a generative model is only capable of generating samples from a subset of all the modes. Closely related is the problem of complete mode coverage [263]. We speak of complete mode coverage when the generative model is capable of sampling from each mode, even when the number of training samples of a mode is very small. In this thesis, we try to solve both these problems using diversity in Chapter 4.

Aim: To propose a new generative model with a disentangled latent space.

Disentanglement was previously discussed as a tool to provide interpretable information for the user. Typically, an increase in disentanglement performance is traded off with generation quality. By making use of the RKM framework [224], we propose generative restricted kernel machines (RKM). The proposed model performs well in both disentanglement and generation quality. This aim is tackled in Chapter 5.

Aim: To propose a generative model that is resistant against contamination in the training data.

In generative modeling, training data is often assumed to be ground truth, therefore outliers can severely degrade the learned representations and performance of trained models. Contamination in the training data results in the encoding of outliers. Consequently, the network generates noisy images when reconstructing out-of-sample extensions. In this thesis, we robustify the generative RKM method by the introduction of a weighting mechanism based on the MCD estimator in Section 6.1.

Aim: To extend the minimum regularized covariance determinant estimator to the kernel domain.

Two potential downsides of MCD-type estimators were previously discussed. The estimators assume that the non-outlying observations are roughly elliptically distributed, which is not always the case. Second, the computation time increases substantially when the number of variables goes up. Both issues are solved in this thesis by introducing the kernel trick in the MCD-framework in Section 6.2.

1.4 Outline of the thesis

We conclude this section with a chapter-by-chapter outline of the thesis. There are five chapters that can be put under the three major themes of this thesis: diversity, robustness, and disentanglement. A schematic overview of the thesis is given in Figure 1.4.1.

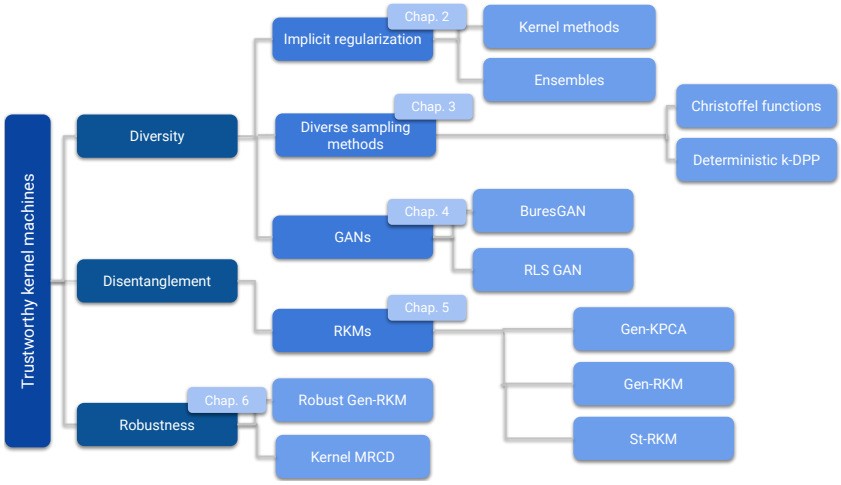


Figure 1.4.1: Overview of the thesis.

Chapter 2 discusses the connection between diversity sampling and implicit regularization. By using DPPs for sampling, we obtain theoretical results concerning the interplay between diversity and regularization. Empirically, we demonstrate that diverse sampling methods improve performance in several kernel methods compared to uniform sampling. The interplay between diversity and implicit regularization is analyzed in the Nyström approximation, KPCA,

KRR, and ensembles of KRR. Moreover, we introduce a loss to measure performance separately in the tail and bulk of the data. The bulk and tail of the data correspond to points with low and high outlyingness respectively, where the outlyingness is measured by the ridge leverage scores. By splitting the loss function into two parts, one can identify if the model is not only focusing on the majority of the data but also performing well in 'outlying' subpopulations.

Chapter 3 introduces new diversity sampling methods. We propose a deterministic and a randomized adaptive algorithm for selecting landmark points within a training data set, which are related to the minima of a sequence of kernelized Christoffel functions. Beyond the known connection between Christoffel functions and *leverage scores*, a connection of our method with *determinantal point processes* is also explained. Namely, our construction promotes diversity among important landmark points in a way similar to DPPs. Also, we explain how our randomized adaptive algorithm can improve the performance of kernel ridge regression, especially in stratified datasets. Besides the previously mentioned algorithms, we discuss a greedy deterministic adaptation of k-DPP. Deterministic algorithms are interesting for many applications, as they provide interpretability to the user by having no failure probability and always returning the same results.

Chapter 4 analyzes the role of diversity in generative adversarial networks. As a solution to mode collapse in GANs, we propose matching the real batch diversity to the fake batch diversity by using the Bures distance between covariance matrices in feature space. The computation of the Bures distance can be conveniently done in either feature space or kernel space in terms of the covariance and kernel matrix respectively. The proposed method decreases mode collapse and improves sample quality. When minority modes are present in the data, a diversity sampling procedure based on ridge leverage scores is proposed. Ridge leverage scores are computed by using an explicit feature map, associated with the next-to-last layer of a GAN discriminator or of a pre-trained network, or by using an implicit feature map corresponding to a Gaussian kernel. Including diversity sampling significantly improves mode coverage when compared to the standard uniform sampling.

Chapter 5 focuses on disentanglement and restricted kernel machines. In this chapter, the generative restricted kernel machine is proposed. We establish connections with probabilistic models, formulate an Evidence Lower Bound (ELBO), and discuss the independence of latent factors. Two training algorithms are proposed: an exact training algorithm for smaller datasets, and a stochastic algorithm for large-scale training. Experiments show the potential of the method in terms of disentanglement and generation quality. Moreover, we demonstrate the use of a disentangled latent space in interpreting the different components on a number of datasets.

Chapter 6 is dedicated to robustness. Two robust methods are introduced. First, we robustify the Gen-RKM method [175]. In this new formulation, the latent space is weighted based on the MCD method. Potential outliers are down-weighted, and therefore not encoded in the representation of the model. Second, the minimum regularized covariance determinant method (MRCD), which is a robust estimator for multivariate location and scatter, is generalized to a kernel-induced feature space. The proposed method allows outlier detection in non-elliptical data when used with a nonlinear kernel.

Chapter 2

Diversity sampling and implicit regularization

This chapter comprises previously published articles including:

- Michaël Fanuel, Joachim Schreurs[1], and Johan AK Suykens. Diversity sampling is an implicit regularization for kernel methods. In *SIAM Journal on Mathematics of Data Science (SIMODS), vol. 3, no. 1, pp. 280-297*, 2021.
- Joachim Schreurs[2], Michaël Fanuel, and Johan AK Suykens. Ensemble Kernel Methods, Implicit Regularization and Determinantal Point Processes. In *ICML workshop on Negative Dependence and Submodularity for ML*, 2020.

[1]The candidate collaborated with the first author on the research, methodology, and the article preparation, with a focus on the experiments.

[2]The candidate performed the research and wrote the article in collaboration with the coauthors.

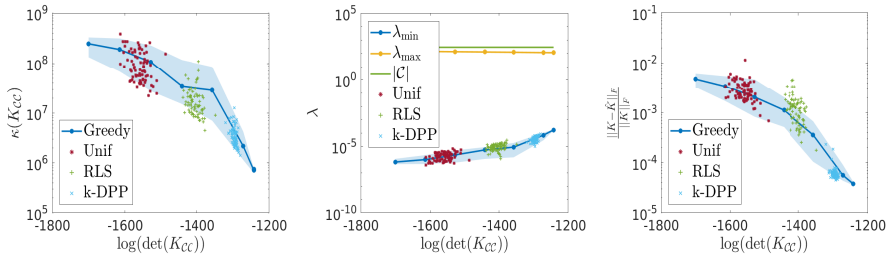
2.1 Diversity sampling is an implicit regularization for kernel methods

2.1.1 Introduction

Kernel methods often rely on low-rank approximations to deal with large-scale datasets. This section addresses the special case of the Nyström approximation that is defined hereafter. Namely, let $k(\boldsymbol{x}, \boldsymbol{y}) > 0$ be a continuous and strictly positive definite kernel. Given data $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i\in[n]}$, kernel methods rely on the entries of the Gram matrix $K = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$. To deal with large scale problems, one often samples a subset of landmarks $\mathcal{C} \subseteq [n]$ and defines a $n \times |\mathcal{C}|$ sampling matrix C obtained by selecting the columns of the identity matrix indexed by $\mathcal{C}$. This is useful to select rectangular and squared submatrices as follows: $K_{\mathcal{C}} = KC$ and $K_{\mathcal{C}\mathcal{C}} = C^\top KC$. Then, the $n \times n$ kernel matrix K is approximated by a low rank Nyström approximation

$$\hat{K}(K, \mathcal{C}) = K_{\mathcal{C}} K_{\mathcal{C}\mathcal{C}}^{-1} K_{\mathcal{C}}^\top, \tag{2.1.1}$$

which involves the inverse of $K_{\mathcal{C}\mathcal{C}}$. However, this submatrix can be ill-conditioned. In practice, this happens especially when $\mathcal{C}$ is sampled uniformly at random and for the Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|_2^2/\sigma^2)$, that is used in this section. We argue here that a sampling of $\mathcal{C}$ which yields a good performance is closely related to the *diversity* of the set of landmarks. In this work, the diversity of $\mathcal{C}$ is measured by the value of $\det(K_{\mathcal{C}\mathcal{C}})$. This is intuitively understood thanks to the connection between determinant and volume [125]. We claim that selecting a diverse sample *implicitly regularizes* the corresponding submatrix. This is illustrated on the `Housing` dataset in Figure 2.1.1 where the Nyström approximation error and condition number of $K_{\mathcal{C}\mathcal{C}}$ are given for several $\mathcal{C}$ of identical cardinality, associated with different diversities. Additional technical details are given in Section 2.1.3. The empirical findings described in Section 2.1.3 indicate that Nyström-based kernel methods are improved if the landmarks are both diverse and yield an accurate kernel approximation. To illustrate this statement, we use a greedy swapping algorithm, namely Algorithm 2.1.1 (blue line in Figure 2.1.1), which allows for obtaining a sample of landmarks with a given diversity. The example of Figure 2.1.1 illustrates the connection between diversity, regularization, and Nyström approximation error. Namely, we sample repeatedly subsets of the same size by using Uniform, Ridge Leverage Score (RLS) [73] and DPP sampling, which yield samples with increasing diversity. Figure 2.1.1 shows that the corresponding kernel submatrices have an increasing least eigenvalue, a decreasing Nyström approximation error, and a decreasing condition number. This highlights the implicit regularization due to diversity. Among diverse sampling methods, DPPs



(a) $\kappa(K_{\mathcal{CC}})$ (b) $\lambda_{\min}(K_{\mathcal{CC}})$ (c) Approximation error

Figure 2.1.1: Nyström approximation for the `Housing` dataset with parameters given in Table A.2.1. The condition number, smallest and largest eigenvalues of $K_{\mathcal{CC}}$, relative Frobenius norm of the approximation error of $\hat{K} = \hat{K}(K, \mathcal{C})$ are plotted versus the sample diversity. The larger $\det(K_{\mathcal{CC}})$, the more diverse the subset. Error bars are the 0.05 and 0.95 quantile over 10 simulations.

provide a natural probabilistic framework for diversity sampling. Their elegant definition allows to derive results formalizing our empirical observations about the interplay between regularization and diversity. Let us briefly define them in the simplest setting, while a more complete overview can be found in [125].

L-ensemble DPP sampling

Let L be a $n \times n$ positive definite symmetric matrix. An L-ensemble, denoted by $DPP_L(L)$, is a DPP with a specific probability mass function for $\mathcal{C} \subseteq [n]$, which depends on the matrix L. Namely, when $\mathcal{Y} \sim DPP_L(L)$, we have

$$\Pr(\mathcal{Y} = \mathcal{C}) = \det(L_{\mathcal{CC}}) / \det(\mathbb{I} + L),$$

where the normalization is designed such that these probabilities sum up to 1 (see [125]). In contrast to (1.2.2), the only requirement on L is that it has to be positive semidefinite. In this section, we define $L = K/\alpha$ where $\alpha > 0$ is a scale parameter that influences the expected subset size. Classically, an alternative viewpoint deals with the inclusion probabilities as given by $\Pr(\mathcal{C} \subseteq \mathcal{Y}) = \det(P_{\mathcal{CC}})$, where

$$P = K(K + \alpha\mathbb{I})^{-1}, \tag{2.1.2}$$

is the marginal kernel corresponding to the L-ensemble with $L = K/\alpha$. The diagonal of the soft projector matrix (2.1.2) yields the Ridge Leverage Scores (RLS) of the data points:

$$\ell_i = P_{ii} \text{ for } i \in [n],$$

where ℓ_i denotes the i-th entry of the vector $\boldsymbol{\ell} \in \mathbb{R}^n$ containing the RLS. These scores have been used in order to sample landmarks points in various works [165, 73, 15] in the context of Nyström approximations. RLS can be considered as a measure of importance or 'outlierness' of a data point. Moreover, connections between RLS, DPPs and Christoffel functions are explored in the next chapter. The sum of the RLS yields the effective dimension $d_{\text{eff}}(K/\alpha)$ which is also the expected size $d_{\text{eff}}(K/\alpha) = \mathbb{E}_{\mathcal{C}}[|\mathcal{C}|]$ if $\mathcal{C} \sim DPP_L(K/\alpha)$. Since the subset size $|\mathcal{C}|$ in itself is also a random variable, it is also customary to use k-DPPs which are DPPs conditioned on a given subset size k (see also [136]).

We now give a short overview of how the Nyström approximation is used to speed up kernel PCA and kernel ridge regression.

Kernel PCA

The Nyström method is used to develop a more computationally efficient approximate kernel PCA algorithm [220]. Let $\mathcal{H}$ be the Reproducing Kernel Hilbert Space associated to k and assume that the data is sampled from a distribution ρ such that $\mathbb{E}_{X\sim\rho}[f(X)] = 0$ for all $f \in \mathcal{H}$. We recall that Kernel PCA is a principal component analysis in a RKHS, i.e., it consists in finding the directions of maximum variance. Indeed, let $k_{\boldsymbol{x}_i}(\cdot) = k(\boldsymbol{x}_i, \cdot)$ and let the empirical covariance operator be $C = \frac{1}{n}\sum_{i=1}^n k_{\boldsymbol{x}_i} \otimes k_{\boldsymbol{x}_i}$. Also, we define the subspace $\mathcal{H}_{\mathcal{C}} = \text{span}\{k_{\boldsymbol{x}_i} \text{ s.t. } i \in \mathcal{C}\}$. Then, the optimization problem

$$\sup_f \langle f, Cf\rangle_{\mathcal{H}} \text{ s.t. } \|f\|_{\mathcal{H}} = 1 \text{ and } f \in \mathcal{H}_{\mathcal{C}},$$

corresponds to a Nyström approximation of KPCA if $\mathcal{C} \subset [n]$. The empirical estimation of KPCA involves the eigendecomposition of the matrix

$$M = K_{\mathcal{C}\mathcal{C}}^{-1/2} K_{\mathcal{C}}^{\top} K_{\mathcal{C}} K_{\mathcal{C}\mathcal{C}}^{-1/2},$$

sharing its non-zero eigenvalues with (2.1.1). Let $(\hat{\lambda}_{\ell,\mathcal{C}}, \boldsymbol{u}_\ell)_{\ell=1}^{|\mathcal{C}|}$ be eigenpairs of M sorted in descending order. KPCA aims to construct the orthogonal projector $P_{\mathcal{C}} = \sum_{\ell=1}^c \boldsymbol{u}_\ell \boldsymbol{u}_\ell^{\top}$ on the subspace corresponding to the c largest eigenvalues, with $c \leq |\mathcal{C}|$. Clearly, the conditioning of $K_{\mathcal{C}\mathcal{C}}$ is important for this task in view of the definition of M. The reconstruction error for c components assesses the quality of the approximation and is given by:

$$\hat{R}(P_{\mathcal{C}}) = \text{tr}(K) - \sum_{\ell=1}^c \boldsymbol{u}_\ell^{\top} M \boldsymbol{u}_\ell = \sum_{\ell=1}^n \hat{\lambda}_\ell - \sum_{\ell=1}^c \hat{\lambda}_{\ell,\mathcal{C}}, \tag{2.1.3}$$

where $(\hat{\lambda}_\ell, \boldsymbol{v}_\ell)_{\ell=1}^n$ are eigenpairs of K. A small reconstruction error is then achieved thanks to an accurate Nyström approximation as detailed in Section 2.1.2.

Regression

In approximate Kernel Ridge Regression (KRR), the regressor is obtained from given input-output pairs $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i\in[n]}$ by solving

$$f^\star = \arg\min_{f\in\mathcal{H}_\mathcal{C}} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2 + \gamma\|f\|_\mathcal{H}^2, \text{ with } \gamma > 0, \tag{2.1.4}$$

where $\mathcal{H}_\mathcal{C} = \text{span}\{k(\boldsymbol{x}_i, \cdot)|i \in \mathcal{C}\}$ and $\mathcal{C} \subseteq [n]$. The regressor is $f^\star(\cdot) = \sum_{i\in\mathcal{C}} \alpha_i^\star k(\boldsymbol{x}_i, \cdot)$ with

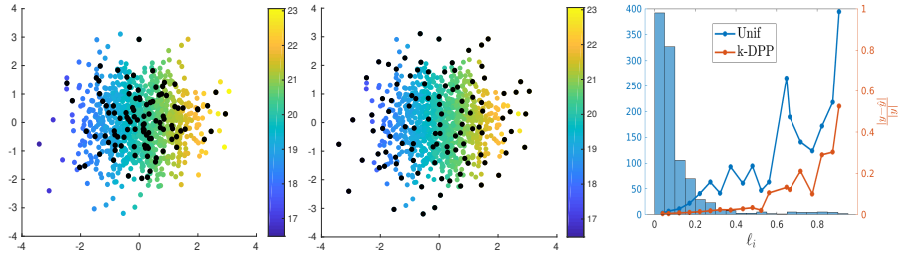
$$\boldsymbol{\alpha}^\star = (K_\mathcal{C}^\top K_\mathcal{C} + n\gamma K_{\mathcal{C}\mathcal{C}})^{-1} K_\mathcal{C}^\top \boldsymbol{y}. \tag{2.1.5}$$

The condition number of (2.1.5) crucially depends on the magnitude of the least eigenvalue of $K_{\mathcal{C}\mathcal{C}}$ which plays the role of regularization term. Notice that the full KRR is simply obtained by replacing $\mathcal{H}_\mathcal{C}$ by $\mathcal{H}$ in (2.1.4).

The following two paragraphs motivate the impact of the regularity of $K_{\mathcal{C}\mathcal{C}}$ in two applications. Firstly, a better kernel approximation yields an improvement of the performance of unsupervised kernel methods such as Kernel Principal Component Analysis (KPCA) [208, 220] and Kernel k-means [240]. Secondly, the conditioning of $K_{\mathcal{C}\mathcal{C}}$ is also important for large-scale supervised learning methods – based on Nyström approximation – as the convergence and accuracy of iterative solvers depends often on the condition number. Finally, sampling with a diverse method spreads the points more over the full dataset. This is especially important for accuracy in less populated or 'outlying' regions in the dataset.

Stability of the expected risk. The first consequence of an accurate Nyström approximation is that the expected risk of approximate KRR is upper bounded by a controllable constant as it is explained in Section 2.1.2. This means that the training problem can not be dramatically affected by the Nyström approximation.

Uniform test error. A second consequence is directly related to the diversity of the landmarks and is illustrated in Figure 2.1.2. Namely, Figure 2.1.2 shows the training dataset which consists of 1000 sampled points generated as follows: $\boldsymbol{x}_i \sim \mathcal{N}(0, \mathbb{I})$ and $y_i = \boldsymbol{e}_1^\top \boldsymbol{x}_i + b + \epsilon_i$ with iid $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = 0.1$ and $b = 20$. The corresponding y_i values are visualized by the color of the points. Landmarks are then sampled by using uniform and DPP sampling. Uniform sampling oversamples the dense parts, while a diverse sampling algorithm spreads the points over the full dataset. A kernel ridge regressor using a



(a) Uniform sampling. (b) k-DPP sampling. (c) RLS and binned MAPE.

Figure 2.1.2: Toy example of regression. In 2.1.2c, a histogram of RLS distribution and the MAPE test error in each bin are displayed.

Gaussian kernel with $\sigma = 0.5$ is trained by using (2.1.4), where the optimal regularization parameter $\gamma > 0$ is determined by using cross-validation. Figure 2.1.2c shows the Mean Absolute Percentage Error (MAPE) as a function of the ridge leverage scores of the test set, where each dot corresponds to the MAPE in the corresponding bin of the histogram. This stratification of the dataset allows to visualize how the regressor performs in dense (small RLS) and sparser (large RLS) groups of the dataset.

Diverse sampling has consistently a better performance with respect to uniform sampling, while the difference is more apparent for high leverage scores. This is especially important when sampling from datasets with long-tail RLS distributions. Hence, in the case of diverse sampling, we emphasize that the percentage error is more uniform on the support of the dataset while the regressor makes a smaller error on points with a larger leverage score compared to regressors obtained with uniform sampling, while the total MAPE shows only a minor difference. Additional illustrations of this effect are given in Section 2.1.3. Naturally, diverse sampling is less important if there is no long tail in the RLS distribution.

The rest of the section includes theoretical results in Section 2.1.2, and numerical experiments in Section 2.1.3. The proofs and dataset description are given in Appendix A. Another application, namely kernel k-means, and additional numerical experiments can be found in Appendix A.3.

2.1.2 Main results: Implicit regularization

On expectation, the largest and smallest eigenvalues of several matrices obtained by DPP sampling can be bounded, showing indeed that the spectrum of those

submatrices is likely to be under control. This is formalized in Theorem 2.1.1, where we denoted by $\circ$ the entry-wise product between matrices.

Theorem 2.1.1 (Implicit regularization)**.** *Let $\mathcal{C} \sim DPP_L(K/\alpha)$ and let C be a sampling matrix associated to the set $\mathcal{C}$. Then, we have*

$$\mathbb{E}_C\left[CK_{\mathcal{C}\mathcal{C}}^{-1}C^\top\right] = (K+\alpha\mathbb{I})^{-1}, \text{ and } \mathbb{E}_C\left[CK_{\mathcal{C}\mathcal{C}}C^\top\right] = P_{(2)} \circ K,$$

with $P_{(2)} = \mathrm{Diag}(\boldsymbol{\ell}) + \boldsymbol{\ell}\boldsymbol{\ell}^\top - P \circ P \succeq 0$.

It is worth remarking that the strength of the expected implicit regularization originates from the scaling of the matrix defining the L-ensemble $DPP_L(K/\alpha)$. Also, notice that $P_{(2)}$ is positive semi-definite. Furthermore, if $\mathrm{diag}(K) = \mathbf{1}$ as in the case of the Gaussian kernel, the largest eigenvalue

$$\lambda_{\max}(\mathbb{E}_C\left[CK_{\mathcal{C}\mathcal{C}}C^\top\right]) \leq \lambda_{\max}(P_{(2)}) \leq \|\boldsymbol{\ell}\|_\infty + \|\boldsymbol{\ell}\|_2^2, \tag{2.1.6}$$

is bounded in terms of the leverage scores. This is a direct consequence of a Corollary 2 in [16], namely the spectrum of $A \circ B$ is majorized by the spectrum of A, if A and B are symmetric and positive semidefinite with $\mathrm{diag}(B) = \mathbf{1}$. It is noticeable that the largest eigenvalue of the expected kernel submatrix is under control for DPP sampling. Indeed, other sampling schemes are not known to yield similar guarantees. Again, if $\mathrm{diag}(K) = \mathbf{1}$, the trace of $\mathbb{E}_{\mathcal{C}}\left[C^\top K_{\mathcal{C}\mathcal{C}}C\right])$ is the expected size of the sample, $d_{\mathrm{eff}}(K/\alpha)$, which gives then an upper bound for $\lambda_{\max}(\mathbb{E}_C\left[CK_{\mathcal{C}\mathcal{C}}C^\top\right])$. We observe empirically that the latter yields a much larger upper bound compared to (2.1.6). Importantly, the scale parameter $\alpha > 0$ both controls the size of the sample and regularizes the subkernels matrix in the following sense:

$$\lambda_{\max}(\mathbb{E}_{\mathcal{C}}\left[CK_{\mathcal{C}\mathcal{C}}^{-1}C^\top\right]) \leq \alpha^{-1},$$

where $\mathcal{C} \sim DPP_L(K/\alpha)$. These results on expectation can be instructive since we expect concentration about the mean. Permantle and Peres showed that strong Rayleigh measures – generalizing DPPs – obey Gauss-Poisson concentration bounds [178]. Corollary 2.1.1 is then a direct consequence of that concentration result. For convenience, we write $\boldsymbol{w}_{\mathcal{C}} = C^\top \boldsymbol{w}$, where C is the sampling matrix associated to $\mathcal{C} \subseteq [n]$.

Corollary 2.1.1 (Regularization with high probability)**.** *Let $\boldsymbol{w} \in \mathbb{R}^n$ a fixed vector such that $\|\boldsymbol{w}\|_2 = 1$. Let $\mathcal{C} \sim DPP_L(K/\alpha)$. Then, we have*

$$\left|\boldsymbol{w}_{\mathcal{C}}^\top K_{\mathcal{C}\mathcal{C}}^{-1}\boldsymbol{w}_{\mathcal{C}} - \boldsymbol{w}^\top(K+\alpha\mathbb{I})^{-1}\boldsymbol{w}\right| \leq \frac{8}{\lambda_{\min}(K)}\left(2+\sqrt{\frac{d_{\mathrm{eff}}(K/\alpha)}{2}}\right)\log\left(\frac{5}{\delta}\right),$$

with probability at least $1-\delta$, where $d_{\mathrm{eff}}(K/\alpha) = \mathrm{Tr}(K(K+\alpha\mathbb{I})^{-1})$ is the expected subset size of $DPP_L(K/\alpha)$.

Note that when α is large, we have large regularization and less points sampled, which results in a tighter bound. A drawback of Corollary 2.1.1 is that the bound hereabove depends on the inverse of $\lambda_{\min}(K)$ which may be a large number. The result may be improved by finding a better upper bound on the Lipschitz constant of the function $f(\mathcal{C}) = \boldsymbol{w}_{\mathcal{C}}^{\top} K_{\mathcal{C}\mathcal{C}}^{-1} \boldsymbol{w}_{\mathcal{C}}$. We refer to the proof of Corollary 2.1.1 for more details.

Nyström approximation

The subset obtained thanks to a DPP sampling is not only yielding a regular kernel submatrix, Corollary 2.1.2 states that it produces a good Nyström approximation. It also gives a natural connection between the projector

$$\mathbb{P}_{\mathrm{range}(K^{1/2}C)} = K^{1/2} C (K_{\mathcal{C}\mathcal{C}})^{-1} C^{\top} K^{1/2},$$

and the marginal kernel (2.1.2).

Corollary 2.1.2 (Expected Nyström approximation)**.** *Let $\mathcal{C} \sim DPP_L(K/\alpha)$. Then, we have an expression for the Nyström error on expectation*

$$\mathbb{E}_{\mathcal{C}} \left[K - \hat{K}(K, \mathcal{C}) \right] = \alpha P, \text{ and } \mathbb{E}_{\mathcal{C}} \left[\mathbb{P}_{\mathrm{range}(K^{1/2}C)} \right] = P.$$

As a straightforward consequence of Corollary 2.1.2, the nuclear norm of the approximation error is simply $\mathbb{E}_{\mathcal{C}}[\mathrm{Tr}(K - \hat{K}(K, \mathcal{C}))] = \alpha d_{\mathrm{eff}}(K/\alpha)$, since $K \succeq \hat{K}(K, \mathcal{C})$. To the best of our knowledge, only a weaker result about the accuracy of the Nyström approximation [136] related to k-DPPs exists in the literature.

Unsupervised kernel methods

Theorem 2.1.2 states that the 'distance' of K to a k-dimensional subspace is well approximated by the 'distance' of $\hat{K}(K, \mathcal{C})$ to the same subspace, on expectation. An analogous result of RLS sampling can be found in [165].

Theorem 2.1.2 (Expected projection-cost preservation)**.** *Let $\mathcal{C} \sim DPP_L(K/\alpha)$ and X an orthogonal projector on a k-dimensional subspace. Denote $\hat{K} = \hat{K}(K, \mathcal{C})$. Then we have*

$$\mathrm{Tr}(K - XKX) \leq \mathbb{E}_{\mathcal{C}}[\mathrm{Tr}(\hat{K} - X\hat{K}X)] + c(\alpha) \leq \mathrm{Tr}(K - XKX) + \min\{\alpha k, c(\alpha)\},$$

where $c(\alpha) = \alpha d_{\mathrm{eff}}(K/\alpha)$.

A direct application of the above theorem is KPCA. Namely, the projector $X^\star$ onto the leading k components is obtained by solving

$$\min_{X \in \Pi_k} \operatorname{Tr}(K - XKX),$$

where Π_k is the set of $n \times n$ projectors of rank k. Then, the result Theorem 2.1.2 is a stability result relating the objective functions of KPCA with and without Nyström approximation.

Kernel ridge regression

A simple consequence of Corollary 2.1.2 is that the expected risk of KRR approximated by Nyström method with DPP sampling cannot be arbitrarily larger than the risk corresponding to the full KRR. Namely, let the observed outputs be $y_i = z_i + \epsilon_i$ where ϵ_i are iid $\mathcal{N}(0, \sigma^2)$ and z_i the noiseless output. Let the solution of KRR be $\hat{\boldsymbol{z}}_K = K(K + n\gamma\mathbb{I})^{-1}\boldsymbol{y}$, and denote by $\hat{\boldsymbol{z}}_{\hat{K}(K,\mathcal{C})}$ the solution obtained by using the Nyström approximation $\hat{K}(K, \mathcal{C})$. The expected risk is then defined as $\mathcal{R}(\hat{\boldsymbol{z}}_K) = \mathbb{E}_\epsilon \|\hat{\boldsymbol{z}}_K - \boldsymbol{z}\|_2^2$. Then, we can give a bound on the risk of KRR associated to the Nyström approximation.

Theorem 2.1.3 (Expected risk bound)**.** *Let $\mathcal{C} \sim DPP_L(K/\alpha)$, then we have*

$$\mathbb{E}_{\mathcal{C}} \left[\sqrt{\frac{\mathcal{R}(\hat{\boldsymbol{z}}_{\hat{K}(K,\mathcal{C})})}{\mathcal{R}(\hat{\boldsymbol{z}}_K)}} \right] \leq 1 + \frac{\alpha}{n\gamma} d_{\text{eff}}(K/\alpha).$$

The upper bound in Theorem 2.1.3 tends to 1 as $\alpha \to 0$ since $d_{\text{eff}}(K/\alpha) \leq n$. This consistently shows that the larger is the number of landmarks, the closest is the risk of approximate KRR from the full KRR. Notice that the increase in the risk is also mitigated by the regularization parameter.

Preconditioners. Rudi et al. [201] propose a preconditioning of the linear system (2.1.5) of the form

$$B^\top (K_{\mathcal{C}}^\top K_{\mathcal{C}} + n\gamma K_{\mathcal{C}\mathcal{C}}) B \big(B^{-1}\boldsymbol{\alpha}\big) = B^\top K_{\mathcal{C}}^\top \boldsymbol{y} \tag{2.1.7}$$

where B is obtained by solving $BB^\top = \big(K_{\mathcal{C}\mathcal{C}} D_{\mathcal{C}\mathcal{C}} K_{\mathcal{C}\mathcal{C}} + n\gamma K_{\mathcal{C}\mathcal{C}}\big)^{-1}$, thanks to a Cholesky decomposition, where $D_{\mathcal{C}\mathcal{C}}$ is an appropriate diagonal matrix. In the case of the uniform sampling of $\mathcal{C}$, the authors of [201] propose $D_{\mathcal{C}\mathcal{C}} = (n/|\mathcal{C}|)\mathbb{I}_{\mathcal{C}\mathcal{C}}$. For RLS sampling, they argue for $D_{\mathcal{C}\mathcal{C}} = \operatorname{Diag}(\boldsymbol{\ell}_{\mathcal{C}})^{-1}$, where $\boldsymbol{\ell}$ contains the so-called ridge leverage scores. We emphasize that the computation of B indeed

crucially depends on the magnitude of the least eigenvalue of $K_{\mathcal{C}\mathcal{C}}$. It is then interesting to sample diverse landmarks so that $K_{\mathcal{C}\mathcal{C}}$ is likely to be regular. A natural motivation for choosing $D_{\mathcal{C}\mathcal{C}} = \mathrm{Diag}(\boldsymbol{\ell}_{\mathcal{C}})^{-1}$ from the DPP viewpoint is given in Corollary 2.1.3. This result naturally follows from Lemma 2.1.1, which can also be found in the context of Monte-Carlo integration [17] with projective DPPs.

Lemma 2.1.1. *Let $\mathcal{C} \sim DPP_L(K/\alpha)$ and $\boldsymbol{v}$ and $\boldsymbol{w} \in \mathbb{R}^n$ Then, we have the identities $\mathbb{E}_{\mathcal{C}}\left[\boldsymbol{v}_{\mathcal{C}}^\top \boldsymbol{w}_{\mathcal{C}}\right] = \boldsymbol{v}^\top \mathrm{Diag}(\boldsymbol{\ell})\boldsymbol{w}$, and $\mathbb{V}_{\mathcal{C}}\left[\boldsymbol{v}_{\mathcal{C}}^\top \boldsymbol{w}_{\mathcal{C}}\right] = (\boldsymbol{v}\circ\boldsymbol{w})^\top\Big(\mathrm{Diag}(\boldsymbol{\ell}) - P\circ P\Big)(\boldsymbol{v}\circ\boldsymbol{w})$.*

Corollary 2.1.3 then motivates the approximation of $K_{\mathcal{C}}^\top K_{\mathcal{C}}$ in (2.1.5) by $K_{\mathcal{C}\mathcal{C}} D_{\mathcal{C}\mathcal{C}} K_{\mathcal{C}\mathcal{C}}$.

Corollary 2.1.3. *Let $\mathcal{C} \sim DPP_L(K/\alpha)$ and $K_{\mathcal{C}} = KC$. Then, the following identity holds: $\mathbb{E}_{\mathcal{C}}\left[K_{\mathcal{C}}\,\mathrm{Diag}(\boldsymbol{\ell}_{\mathcal{C}})^{-1}K_{\mathcal{C}}^\top\right] = K^2$.*

A formula for the variance can also be obtained thanks to Lemma 2.1.1. Again, using DPP sampling with inverse leverage score preconditioning has the advantage that $K_{\mathcal{C}\mathcal{C}}$ is likely to be regular, in contrast with leverage score sampling.

2.1.3 Experimental results

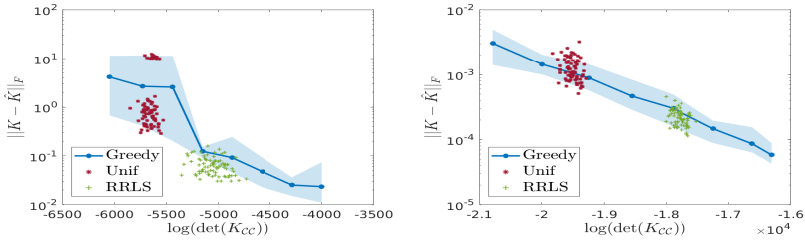
In this section, we illustrate the effect of sampling a subset $\mathcal{C}$ with small or large $\det(K_{\mathcal{C}\mathcal{C}})$ on a number of public datasets. A swapping algorithm, described in Algorithm 2.1.1, is used to determine subsets of size $|\mathcal{C}| = k$ with a given log-determinant d_p, such that $|\log(\det(K_{\mathcal{C}\mathcal{C}})) - d_p| \leq \epsilon$ and where $\epsilon > 0$ is a numerical tolerance. The method swaps points in and out of an initial subset so that the swapped point is accepted if the determinant of the new submatrix is closer to the desired determinant d_p. If the determinant of the subset is too small, we sample a new candidate by using (approximate) leverage scores sampling. Otherwise, if the determinant is too large, we use inverse leverage scores sampling. The size of the subset is chosen to be the effective dimension $k = \sum_{i=1}^n \ell_i$, where $\{\ell_i\}_{i=1}^n$ are the ridge leverage scores with regularization parameter $\alpha = \lambda n$. The same ridge leverage scores are used in the greedy swapping algorithm. The algorithm stops if the desired precision is reached or the number of iterations exceeds 2000, whichever happens, earlier. For large-scale problems, the ridge leverage scores are approximated using Recursive Ridge Leverage Sampling (RRLS) [165] with n_{RRLS} points. The size of the subset is chosen to be the effective dimension of the approximate RLS with

Algorithm 2.1.1 Greedy Swapping Algorithm based on the (approximated) ridge leverage scores.

input: Matrix $K \succ 0$, sample size k, ridge leverage scores $\{\ell_i\}_{i=1}^n$, preferred log-determinant d_p and precision $\epsilon > 0$.
initialization: Sample an initial subset $|\mathcal{C}| = k$ uniformly at random.
Determine the Cholesky decomposition R, with $K_{\mathcal{C}\mathcal{C}} = R^\top R$.
repeat:
Determine the log-determinant $d = 2\sum_{i=1}^k \log(R_{ii})$.
if: $|d - d_p| \leq \epsilon$
break
if: $d < d_p$
Sample a new point $\tilde{c}$ out of the remaining subset with $p_i \sim \ell_i$
otherwise:
Sample a new point $\tilde{c}$ out of the remaining subset with $p_i \sim 1 - \ell_i$
Swap a uniform selected point out of $\mathcal{C}$ with the newly sampled $\tilde{c}$, which gives the new subset $\tilde{\mathcal{C}}$
Do a rank-1 update to the Cholesky decomposition, which gives $\tilde{R}$ and determine $\tilde{d} = 2\sum_{i=1}^k \log(\tilde{R}_{ii})$.
if: $|\tilde{d} - d_p| \leq |d - d_p|$
Keep the swapped point and update $R = \tilde{R}$, $\mathcal{C} = \tilde{\mathcal{C}}$
return $\mathcal{C}$.

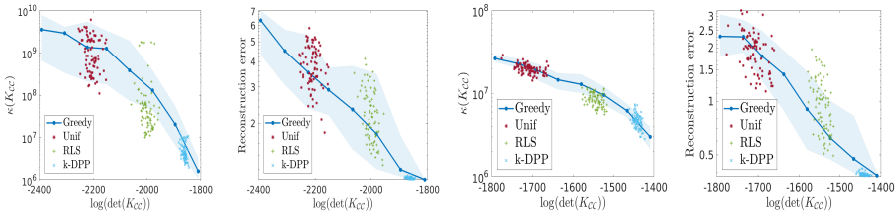
regularization parameter $\alpha = \lambda n$. The same approximate RLSs are used in the greedy swapping algorithm, where the maximum number of iterations is set to 5000. It is worth mentioning that a faster approximation of RLS can be obtained by BLESS [200], although it requires a bit more parameter tuning compared with RRLS. This is why the latter method has been selected for RLS sampling in this section.

Settings. In the sequel, a Gaussian kernel with bandwidth σ is used after standardizing the data. All the simulations are repeated 10 times, the averaged is displayed and the error bars show the 0.05 and 0.95 quantile. The datasets and hyperparameters are given in Table A.2.1. In the first case studies, the following exact algorithms are used to sample k landmarks: Uniform sampling (Unif.), Ridge Leverage Score sampling (RLS) [73] and k-DPP [124]. For a fair comparison, we chose to use k-DPP rather than DPP so that the number of landmarks is always constant and equal to the expected subset size of the associated DPP. In the large-scale experiments: Unif., RRLS, and the greedy swapping method are compared.



(a) `MiniBooNE`: error (b) `codRNA`: error

Figure 2.1.3: Large-scale Nyström approximation results. The Frobenius norm of the approximation error of $\hat{K} = \hat{K}(K, \mathcal{C})$ is plotted versus the sample diversity. The larger $\det(K_{\mathcal{CC}})$, the more diverse the subset.



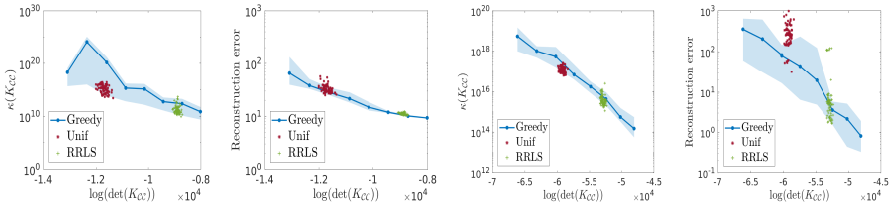
(a) `A.Credit`: $\kappa(K_{\mathcal{CC}})$ (b) `A.Credit`: error (c) `B.Cancer`: $\kappa(K_{\mathcal{CC}})$ (d) `B.Cancer`: error

Figure 2.1.4: KPCA results. The condition number, minimum eigenvalues and reconstruction error using half of the components are plotted as a function of $\det(K_{\mathcal{CC}})$.

Nyström approximation. The impact of diversity on the Nyström approximation is illustrated on the `Housing`, `Abalone`, `codRNA` and `MiniBooNE` datasets[1]. The condition number of $K_{\mathcal{CC}}$, its largest/smallest eigenvalues, and the accuracy of the Nyström approximation are plotted as a function of the determinant in Figures 2.1.1 and 2.1.3.

For completeness, the largest and smallest eigenvalues of $K_{\mathcal{CC}}$ are also given in Appendix A.3. The accuracy of the approximation is evaluated by calculating $\|K - \hat{K}\|_F / \|K\|_F$ with $\hat{K} = KC(K_{\mathcal{CC}} + \varepsilon \mathbb{I}_{\mathcal{CC}})^{-1} C^\top K$ with $\varepsilon = 10^{-12}$ for numerical stability. Afterwards, the following algorithms are used to sample k landmarks: Uniform sampling (Unif.), exact ridge leverage score sampling (RLS) [73] and k-DPP [124]. The results in Figure 2.1.1 show that the 3 sampling algorithms follow the general trend of the greedy swapping algorithm, namely, we have to following empirical observations: 1) Sampling a more diverse subset

[1] `https://www.cs.toronto.edu/~delve/data/datasets.html`, `https://archive.ics.uci.edu/ml/index.php`



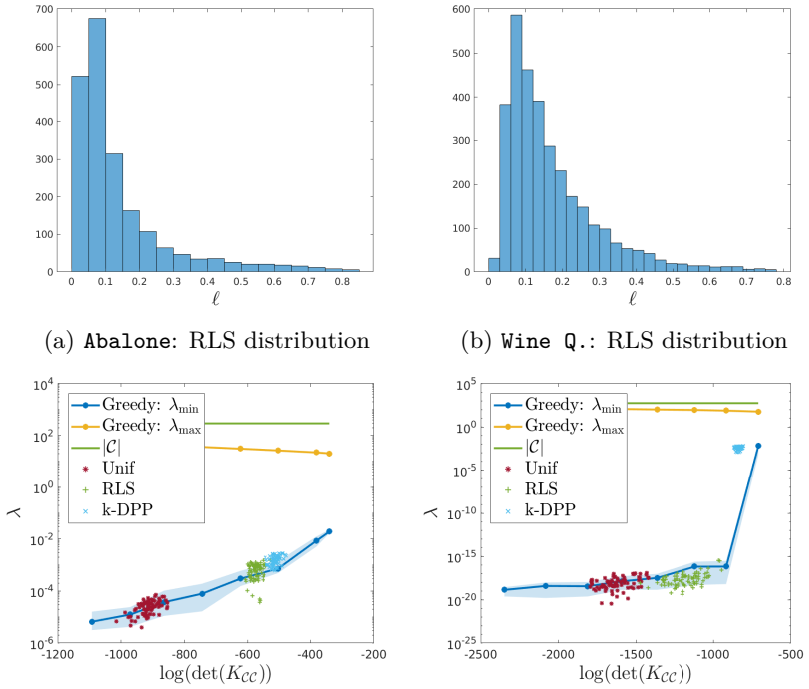
(a) `Adult`: $\kappa(K_{CC})$ (b) `Adult`: error (c) `Cov`: $\kappa(K_{CC})$ (d) `Cov`: error

Figure 2.1.5: Large-scale KPCA results. The condition number and reconstruction error using half of the components are plotted as a function of $\det(K_{CC})$.

results in a smaller condition number $\kappa(K_{CC})$. This is mainly because a larger determinant corresponds to a larger $\lambda_{\min}(K_{CC})$. Indeed, diverse sampling is a computational regularization. 2) Sampling a diverse subset gives a more accurate Nyström approximation. In practice, we observe that RLS sampling yields *effectively* more diverse samples compared to uniform sampling. Notice that in the presence of outliers, taking samples with an extremely large $\det(K_{CC})$ thanks to the Greedy Swapping Algorithm might increase the error on the Nyström approximation as it explained in Appendix A.3. The results for the large-scale experiments are visualized in Figure 2.1.3. The accuracy of the approximation is now evaluated by averaging the Frobenius norm error $\|K - \hat{K}\|_F$ over 50 subsets of size 3000.

Kernel PCA. The numerical experiments are done on the `Breast Cancer` (`B. Cancer`), `Australian Credit` (`A. Credit`), `Adult` and `Covertype` datasets[3]. The condition number, smallest/largest eigenvalues of K_{CC} and the reconstruction error are plotted as a function of the determinant. The averaged results are visualized in Figure 2.1.4. Information about the datasets and hyperparameters used for the experiments is given in Table A.2.1 in the Appendix. Empirically, sampling more diverse subsets results in a smaller reconstruction error and a smaller condition number. The results for the large-scale experiments are visualized on Figure 2.1.5.

Regression and Stratification of the error. To conclude, we verify the usefulness of diversity for a supervised learning task. The dataset is split into 50% training data and 50% test data, so to make sure the train and test set have similar RLS distributions. The test RLS distribution is visualized in Figure 2.1.6. The regression experiment is repeated on the `Abalone`, `Wine Quality`, `Bike Sharing` (`Bike S.`) and `YearPredictionMSD` (`Year`) datasets[3]



(a) **Abalone**: RLS distribution

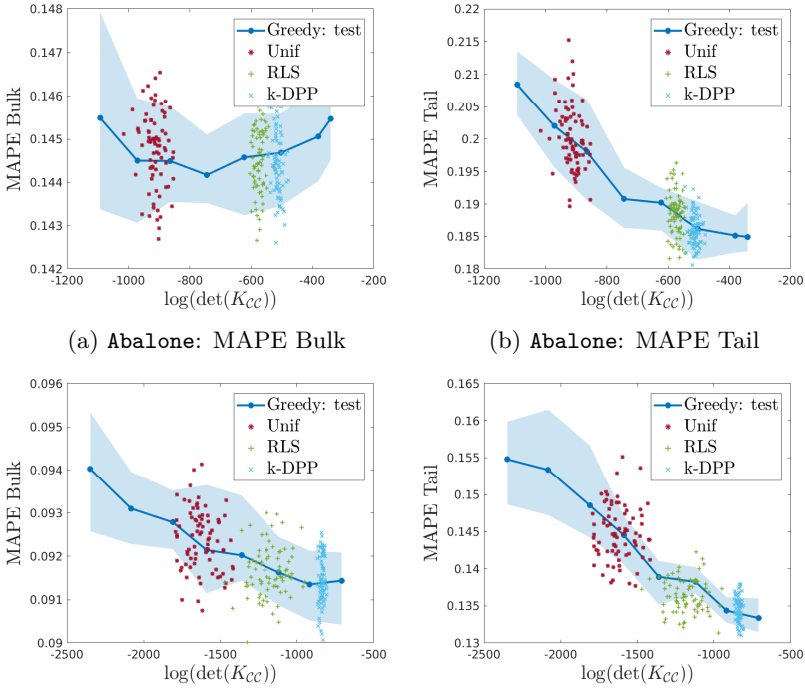
(b) **Wine Q.**: RLS distribution

(c) **Abalone**: eigenvalues

(d) **Wine Q.**: eigenvalues

Figure 2.1.6: Regression results. The histogram of the RLS of the test set and minimum and maximum eigenvalues of $K_{\mathcal{CC}}$ versus $\det(K_{\mathcal{CC}})$, accompanying Figure 2.1.7.

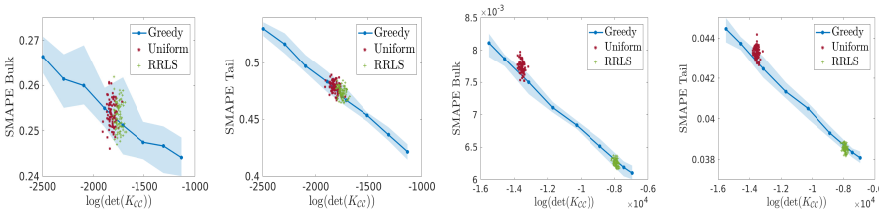
by using KRR. The MAPE of the kernel ridge regression is calculated as a function of $\det(K_{\mathcal{CC}})$. To evaluate the performance, the dataset is stratified, i.e., the test set is divided into 'bulk' and 'tail' as follows: the bulk corresponds to test points where the RLS are smaller than or equal to the 70% quantile, while the tail of the data corresponds to test points where the ridge leverage score is larger than the 70% quantile. The regularization parameter γ is determined by using cross-validation. The results in Figure 2.1.7 show that the 3 sampling algorithms follow the general trend of the greedy swapping algorithm. Again here, sampling a more diverse subset, results in better conditioning of the kernel sub-matrix. Diverse sampling has comparable performance for the bulk data while performing much better in the tail of the data. This confirms the expectations from the regression toy example. The results for the large-scale experiments are visualized in Figure 2.1.8.



(a) `Abalone`: MAPE Bulk (b) `Abalone`: MAPE Tail

(c) `Wine Q.`: MAPE Bulk (d) `Wine Q.`: MAPE Tail

Figure 2.1.7: Regression results. The MAPE on the test set is plotted as a function of $\det(K_{CC})$. A small MAPE corresponds to a large accuracy.

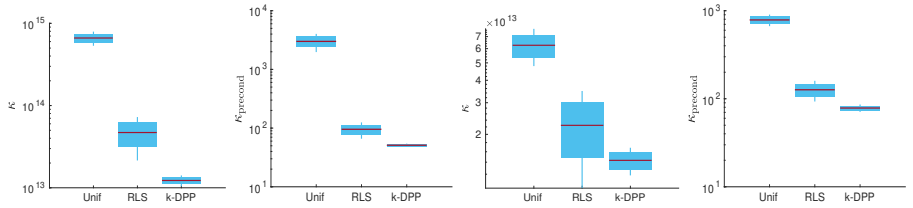


(a) `Bike S.`: Bulk (b) `Bike S.`: Tail (c) `Year Pred.`: Bulk (d) `Year Pred.`: Tail

Figure 2.1.8: Large-scale Regression results. The SMAPE on the bulk and tail of the test set versus the sample diversity. The larger $\det(K_{CC})$, the more diverse the subset.

Preconditioner. A natural idea is to consider iterative methods to solve the system in (2.1.5) because of their simplicity and low iteration cost. The speed and accuracy of convergence of the conjugate gradient method depend on the

condition number of the linear system [119], which makes the use of diverse samplings in combination with iterative methods particularly interesting. This is illustrated on the `Parkinson`, and `Pumadyn8FM` datasets[3]. The condition



(a) `Parkinson` (b) `Parkinson` (c) `Pumadyn8FM` (d) `Pumadyn8FM`

Figure 2.1.9: Preconditioning results. The condition number of the linear system before and after the preconditioning is plotted for Uniform, RLS and k-DPP sampling. From left to right, the condition number before and after preconditioning, for `Parkinson` and `Pumadyn8FM` datasets, respectively.

number of the preconditioned system is computed, where the preconditioner given in (2.1.7) is used in combination with uniform sampling, RLS and k-DPP in combination with the preconditioner defined in (2.1.7). The ridge regularization parameter is equal to $\lambda = 10^{-10}$ to illustrate the impact of diversity on badly conditioned systems. From the results in Figure 2.1.9, we see empirically that sampling a more diverse subset, results in better conditioning of the linear system.

2.2 Ensemble kernel methods, implicit regularization and determinantal point processes

2.2.1 Introduction

The nice theoretical properties of DPPs sparked the search for efficient sampling algorithms. This resulted in many alternative algorithms for DPPs to reduce the computational cost of preprocessing and/or sampling, including many approximate and heuristic approaches. Some examples are the exact sampler without eigendecomposition of [130, 182], coreDPP of [135], or the DPP-VFX algorithm of [58]. The computational cost is often split into two parts: an initial preprocessing cost and subsequent sampling cost. The latter is typically smaller, which makes the previously mentioned algorithms especially useful for sampling multiple small subsets from a large dataset.

We extend the work of the previous section, where the role of *diversity* within kernel methods was investigated. Namely, a more diverse subset results in implicit regularization, which in turn improves the performance of different kernel applications. More specifically we generalize the implicit regularization of DPPs to kDPPs, which are DPPs conditioned on a fixed subset size k [124]. Furthermore, we leverage the common setup of state-of-the-art DPP sampling algorithms, to sample multiple small subsets and use them in an ensemble approach. One can loosely characterize these ensemble approaches as methods wherein the data points are divided into smaller subsets, and estimators are trained on the divisions. Their use has shown to improve performance in Nyström approximation [126] and kernel ridge regression [261, 107, 140]. Experiments show a reduction in error when combining multiple diverse subsets.

Ridgeless kernel regression. Given input-output pairs $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i\in[n]}$, we propose to solve

$$f^\star_{\mathcal{C}} = \arg\min_{f\in\mathcal{H}} \|f\|^2_{\mathcal{H}}, \text{ s.t. } y_i = f(\boldsymbol{x}_i) \text{ for all } i \in \mathcal{C}, \tag{2.2.1}$$

where $\mathcal{C} \subseteq [n]$ is sampled by using a DPP. Here, $\mathcal{H}$ is the reproducing kernel Hilbert space associated with k. The expression of the solution is $f^\star_{\mathcal{C}}(\boldsymbol{x}) = \boldsymbol{k}_x^\top C K_{\mathcal{C}\mathcal{C}}^{-1} C^\top \boldsymbol{y}$, where $\boldsymbol{k}_x = [k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_n)]^\top$. This approximation assumes that some data points can be omitted, contrary to Nyström approximation to Kernel Ridge Regression (KRR) which uses all data points. We show in this section that averaging ridgeless regressors yield the solution of a regularized Kernel Ridge Regression calculated over the complete dataset. For $\mathcal{C} \sim DPP(K/\alpha)$, the expectation of the ridgeless predictors (cfr. Theorem 2.1.1) gives the function

$$\mathbb{E}_C[f^\star_{\mathcal{C}}(\boldsymbol{x})] = \boldsymbol{k}_x^\top (K + \alpha\mathbb{I})^{-1} \boldsymbol{y} =: f^\star(\boldsymbol{x}) \tag{2.2.2}$$

which is the solution of Kernel Ridge Regression with a ridge parameter associated to α, namely

$$f^\star = \arg\min_{f\in\mathcal{H}} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 + \alpha \|f\|^2_{\mathcal{H}}.$$

Typically, a large $\alpha > 0$ yields a small expected subset size for $DPP(K/\alpha)$. In light of the expectation result of (2.2.2), we propose to sample multiple subsets using a DPP and average the ridgeless predictors in an ensemble approach: $\bar{f} = \frac{1}{m}\sum_{i=1}^{m} f^\star_{\mathcal{C}_i}$ with m the number of ensembles. Note that there is no need regularize the KKR itself as sampling a DPP results in implicit regularization on expectation.

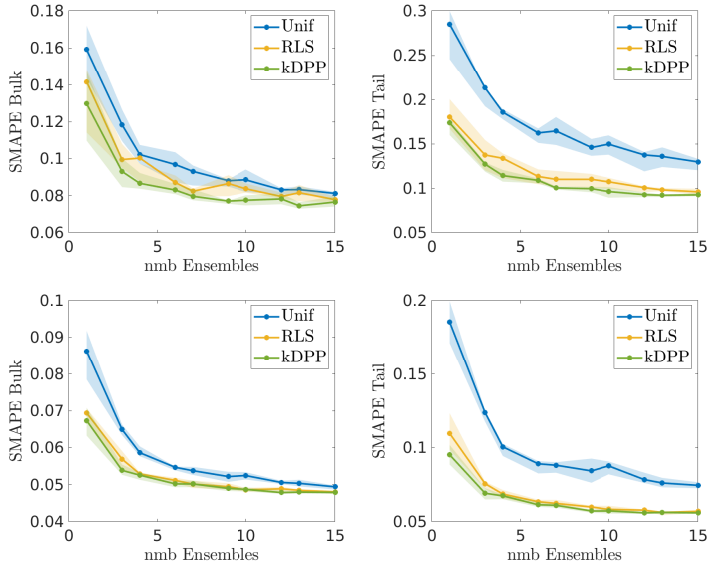


Figure 2.2.1: Ensemble KRR on the Abalone and Wine Quality dataset (from left to right).The SMAPE on the bulk and tail of the dataset is given in function of the number of ensembles.

2.2.2 The analogous result for kDPP sampling

We now discuss an analogous result of Theorem 2.1.1 in the case of kDPPs, for which the implicit regularization effect can be observed. The elementary symmetric polynomial $e_k(K)$ is proportional to the $(n-k)$-th coefficients of the characteristic polynomial $\det(t\mathbb{I} - K) = \sum_{k=0}^{n}(-1)^k e_k(K) t^{n-k}$. Those polynomials are defined on the vector $\boldsymbol{\lambda}$ of eigenvalues of K. There are explicitly given by the formula $e_k(\boldsymbol{\lambda}) = \sum_{1 \leq i_1 < \cdots < i_k \leq n} \lambda_{i_1} \ldots \lambda_{i_k}$. The kDPPs(K) are defined by the subset probabilities $\Pr(Y = \mathcal{C}) = \det(K_{\mathcal{C}\mathcal{C}})/e_k(K) = \det(K_{\mathcal{C}\mathcal{C}})/\sum_{|\tilde{C}|=k} \det(K_{\tilde{C}\tilde{C}})$, and corresponds to DPPs conditioned to a fixed subset size k. Now, we state a result analogous to Theorem 2.1.1.

Lemma 2.2.1. *Let $\mathcal{C} \sim kDPP(K)$ and $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{R}^n$. We have the identities*

$$\mathbb{E}_C[\boldsymbol{u}^\top C K_{\mathcal{C}\mathcal{C}}^{-1} C^\top \boldsymbol{w}] = \frac{e_k(K) - e_k(K - \boldsymbol{w}\boldsymbol{u}^\top)}{e_k(K)}$$

$$= \frac{(-1)^{k+1}}{(n-k)!} \frac{\mathrm{d}^{(n-k)}}{\mathrm{d}\, t^{n-k}} \left[\frac{\boldsymbol{u}^\top \operatorname{adj}(t\mathbb{I} - K)\boldsymbol{w}}{e_k(K)} \right]_{t=0},$$

where adj *is the adjugate of a matrix.*

Proof. Firstly, we use the matrix determinant lemma:

$$\boldsymbol{u}^\top C K_{\mathcal{C}\mathcal{C}}^{-1} C^\top \boldsymbol{w} = \frac{\det(K_{\mathcal{C}\mathcal{C}}) - \det(K_{\mathcal{C}\mathcal{C}} - C^\top \boldsymbol{w}\boldsymbol{u}^\top C)}{\det(K_{\mathcal{C}\mathcal{C}})}.$$

By taking the expectation over $\mathcal{C} \sim \mathrm{kDPP}(K)$, we find

$$\mathbb{E}[\boldsymbol{u}^\top C K_{\mathcal{C}\mathcal{C}}^{-1} C^\top \boldsymbol{w}] = \frac{e_k(K) - e_k(K - \boldsymbol{w}\boldsymbol{u}^\top)}{e_k(K)} =: \mathcal{E}.$$

where we used that $\sum_{|\mathcal{C}|=k} \det A_{\mathcal{C}\mathcal{C}} = e_k(A)$ for any square matrix A. Next, we use the identity $e_k(K) = \frac{(-1)^k}{(n-k)!}\frac{\mathrm{d}^{(n-k)}}{\mathrm{d}\,t^{n-k}}[\det(t\mathbb{I} - K)]_{t=0}$ to obtain the corresponding coefficient of the polynomial $\det(t\mathbb{I} - K) = \sum_{k=0}^n (-1)^k e_k(K) t^{n-k}$. Then, we use once more the matrix determinant lemma with the matrix $(t\mathbb{I} - K)$ this time. This gives

$$\begin{aligned}
\mathcal{E} &= \frac{(-1)^k}{(n-k)!}\frac{\mathrm{d}^{(n-k)}}{\mathrm{d}\,t^{n-k}}\left[\frac{\det(t\mathbb{I} - K) - \det((t\mathbb{I} - K) - \boldsymbol{w}\boldsymbol{u}^\top)}{e_k(K)}\right]_{t=0} \\
&= \frac{(-1)^k}{(n-k)!}\frac{\mathrm{d}^{(n-k)}}{\mathrm{d}\,t^{n-k}}\left[\frac{\det(t\mathbb{I} - K) - \det(t\mathbb{I} - K)(1 + \boldsymbol{u}^\top (t\mathbb{I} - K)^{-1}\boldsymbol{w})}{e_k(K)}\right]_{t=0} \\
&= \frac{(-1)^{k+1}}{(n-k)!}\frac{\mathrm{d}^{(n-k)}}{\mathrm{d}\,t^{n-k}}\left[\frac{\boldsymbol{u}^\top (t\mathbb{I} - K)^{-1}\boldsymbol{w}}{e_k(K)}\det(t\mathbb{I} - K)\right]_{t=0}.
\end{aligned}$$

Finally, we recall that $\mathrm{adj}(A) = \det(A)A^{-1}$, which completes the proof. □

The implicit regularization due to the diverse sampling is not explicit in Lemma 2.2.1. In order to clarify this formula, we write first an equivalent expression for it. Let the eigendecomposition of K be $K = \sum_{\ell=1}^n \lambda_\ell \boldsymbol{v}_\ell \boldsymbol{v}_\ell^\top$. Denote by $\boldsymbol{\lambda} \in \mathbb{R}^n$ the vector containing the eigenvalues of K, sorted such that $\lambda_1 \geq \cdots \geq \lambda_n$. Let $\boldsymbol{\lambda}_{\hat{k}} \in \mathbb{R}^{n-1}$ be the same vector with λ_k missing.

Corollary 2.2.1. *Let $\mathcal{C} \sim kDPP(K)$. We have the identity:*

$$\mathbb{E}_C[C K_{\mathcal{C}\mathcal{C}}^{-1} C^\top] = \sum_{\ell=1}^n \frac{\boldsymbol{v}_\ell \boldsymbol{v}_\ell^\top}{\lambda_\ell + \frac{e_k(\boldsymbol{\lambda}_{\hat{\ell}})}{e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}})}}. \tag{2.2.3}$$

Proof. To begin with, we expand the adjugate in Lemma 2.2.1 in the basis of eigenvectors of K. This gives

$$\operatorname{adj}(t\mathbb{I} - K) = \det(t\mathbb{I} - K)(t\mathbb{I} - K)^{-1} = \sum_{\ell=1}^{n} \frac{\prod_{\ell'=1}^{n}(t - \lambda_{\ell'})}{t - \lambda_\ell} \boldsymbol{v}_\ell \boldsymbol{v}_\ell^\top$$

Then, by the definition of the polynomials e_k and by noting that $n - k = n - 1 - (k - 1)$, we find

$$\frac{(-1)^{k-1}}{(n-k)!} \frac{\mathrm{d}^{(n-k)}}{\mathrm{d}\, t^{n-k}} \left[\prod_{\ell' \neq \ell} (t - \lambda_{\ell'}) \right]_{t=0} = e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}}),$$

where $\boldsymbol{\lambda}_{\hat{\ell}} \in \mathbb{R}^{n-1}$ is the vector $\boldsymbol{\lambda} \in \mathbb{R}^n$ with λ_ℓ missing. This yields $\mathbb{E}_C[CK_{\mathcal{C}\mathcal{C}}^{-1}C^\top] = \sum_{\ell=1}^{n} \frac{e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}})}{e_k(\boldsymbol{\lambda})} \boldsymbol{v}_\ell \boldsymbol{v}_\ell^\top$. The final identity is obtained by using the following recurrence relation $e_k(\boldsymbol{\lambda}) = \lambda_\ell e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}}) + e_k(\boldsymbol{\lambda}_{\hat{\ell}})$, and $\frac{e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}})}{\lambda_\ell e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}}) + e_k(\boldsymbol{\lambda}_{\hat{\ell}})} = \frac{1}{\lambda_\ell + \frac{e_k(\boldsymbol{\lambda}_{\hat{\ell}})}{e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}})}}$. □

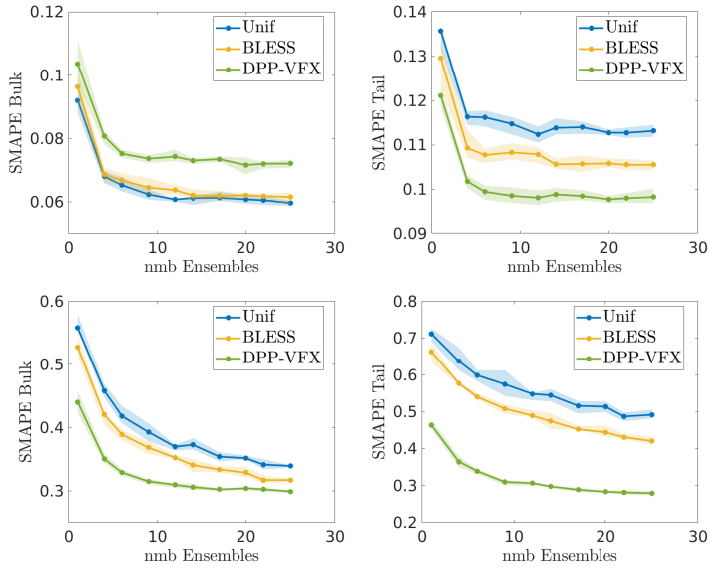


Figure 2.2.2: Ensemble KRR on the Bikesharing and CASP dataset (from left to right). The SMAPE on the bulk and tail of the dataset is given in function of the number of ensembles.

It is now possible to illustrate the connection between Corollary 2.2.1 and implicit regularization. We give a lower bound for the identity in Corollary 2.2.1.

Proposition 2.2.1. *With the notations defined above, we have*

$$\mathbb{E}_C[CK_{\mathcal{C}\mathcal{C}}^{-1}C^\top] \succeq \sum_{\ell=1}^{n} \frac{\boldsymbol{v}_\ell \boldsymbol{v}_\ell^\top}{\lambda_\ell + \alpha}, \tag{2.2.4}$$

where $\alpha = \sum_{i=k}^{n} \lambda_i$ *and* $\mathcal{C} \sim kDPP(K)$.

The above bound matches the expectation formula for DPPs for this specific α. Also, notice that it was remarked in [57] that if $\alpha = \sum_{i=k}^{n} \lambda_i$ then $\mathbb{E}_{\mathcal{C}\sim DPP(K/\alpha)}[|\mathcal{C}|] \leq k$. The inequality (2.2.4) is obtained thanks to the following Lemma with $l = k$.

Lemma 2.2.2 (Eqn 1.3 in [92])**.** *Let* $\boldsymbol{\sigma} \in \mathbb{R}^n$ *be a vector with entries* $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. *Let* k *and* l *be integers such that* $k \geq l > 0$. *Then, we have* $\frac{e_{k+1}(\boldsymbol{\sigma})}{e_k(\boldsymbol{\sigma})} \leq \frac{1}{k-l+1} \sum_{i=l+1}^{n} \sigma_i$.

With the help of Lemma 2.2.2, we can prove (2.2.4).

Proof of Proposition 2.2.1. Let $k \geq 1$. We can lower bound the ratio $\frac{e_{k-1}(\boldsymbol{\lambda}_{\hat{\ell}})}{e_k(\boldsymbol{\lambda}_{\hat{\ell}})}$ in (2.2.3) by using Lemma 2.2.2. Namely let $\boldsymbol{\sigma}$ be the vector $\boldsymbol{\lambda}_{\hat{\ell}} \in \mathbb{R}^{n-1}$ with entries sorted in decreasing order, and let $l = k$. Then, it holds that $\frac{e_k(\boldsymbol{\sigma})}{e_{k-1}(\boldsymbol{\sigma})} \leq \sum_{i=k}^{n-1} \sigma_i$. By using the definition of $\boldsymbol{\sigma}$, we find that, if $k < \ell$, we have $\sum_{i=k}^{n-1} \sigma_i = -\lambda_\ell + \sum_{i=k}^{n} \lambda_i$. Otherwise, if $k \geq \ell$, we have $\sum_{i=k}^{n-1} \sigma_i = \sum_{i=k+1}^{n} \lambda_i$. Hence, we find the upper bound

$$\frac{e_k(\boldsymbol{\sigma})}{e_{k-1}(\boldsymbol{\sigma})} \leq \sum_{i=k}^{n-1} \sigma_i \leq \sum_{i=k}^{n} \lambda_i = \alpha,$$

since $\boldsymbol{\lambda} \geq 0$. Finally, the statement is proved by using the latter inequality and the identity (2.2.3). □

Remark 2.2.1 (Upper bound)**.** *Consider the term* $\ell = n$ *in* (2.2.3). *Then, the additional term at the denominator can be lower bounded as follows:*

$$\frac{e_k(\boldsymbol{\lambda}_{\hat{n}})}{e_{k-1}(\boldsymbol{\lambda}_{\hat{n}})} \geq \frac{n-k}{k} \lambda_{n-1} \left(\frac{\lambda_{n-1}}{\lambda_1}\right)^{k-1} \geq 0,$$

where we used that $e_k(\boldsymbol{\lambda}_{\hat{n}})$ *includes* $\binom{n-1}{k}$ *terms. This bound is pessimistic although it instructs that a small* k *benefits to the regularization.*

As we have observed, the formulae of Theorem 2.1.1 or Corollary 2.2.1 show that the expectation over diverse subsets implicitly regularizes the inverse of the kernel matrix. The improvement of this bound is worth further investigation. A related work [166] uses the same formula given in Theorem 2.1.1 to study the convergence of a random block coordinate optimization method for Kernel Ridge Regression but does not study the ridgeless limit.

2.2.3 Experimental results

Sampling a more diverse subset improves the performance of Nyström approximation and KRR (see previous section). In these experiments, we discuss ensemble approaches for the ridgeless case. The following datasets[2] are used: `Adult`, `Abalone`, `Wine Quality`, `Bike Sharing` and `CASP`. We use 3 sampling algorithms with increasing diversity: uniform sampling, exact ridge leverage score sampling (RLS) [73] and kDPP sampling [124]. For larger datasets, the BLESS algorithm [200] is used instead of RLS and DPP-VFX [58] to speed up the sampling of a kDPP. These algorithms have a relativity small re-sampling cost that motivates their use for ensemble approaches. RLS can be seen as a cheaper proxy for DPP sampling as done in [57]. The different parameters and sample sizes are given in the Supplementary Material. A Gaussian kernel with bandwidth σ is used after standardizing the data. All the simulations are repeated 10 times, the averaged is displayed and the error bars show the 0.25 and 0.75 quantile.

Ensemble Nyström. The accuracy of the approximation is evaluated by calculating $\|K - \hat{K}\|_F / \|K\|_F$ with the ensemble Nyström approximation $\hat{K} = \frac{1}{m}\sum_{i=1}^{m} KC_i(K_{\mathcal{C}_i\mathcal{C}_i} + \varepsilon \mathbb{I}_k)^{-1} C_i^\top K$ with $\varepsilon = 10^{-12}$ for numerical stability. We illustrate the use of diverse ensembles on Figure 2.2.3. Averaging multiple Nyström approximations improve the accuracy. The gain is the most apparent for the more diverse sampling algorithms. Similar to the experiments in [126], we see that uniform sampling combined with equal mixture weights does not improve performance. This is not the case when using more sophisticated sampling algorithms.

Ensemble KRR. Following the implicit regularization of DPP samplings, we asses the performance of averaging ridgeless predictors trained on DPP subsets. Prediction is done by averaging the ridgeless predictors in an ensemble approach: $\bar{f} = \frac{1}{m}\sum_{i=1}^{m} f^\star_{\mathcal{C}_i}$. We evaluate by the same procedure as in the previous section.

[2] `https://archive.ics.uci.edu/ml/index.php`

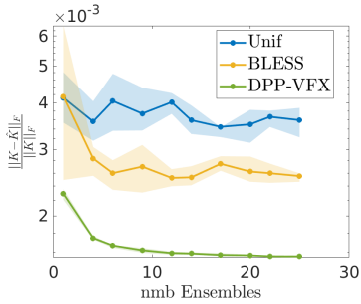


Figure 2.2.3: Ensemble Nyström approximation on the Adult dataset. The relative Forbenius norm of the approximation is given in function of the number of ensembles.

The dataset is split into 50% training data and 50% test data, so to make sure the train and test set have similar RLS distributions. To evaluate the performance, the dataset is stratified, i.e., the test set is divided into 'bulk' and 'tail' as follows: the bulk corresponds to test points where the RLS with regularization $\alpha = 10^{-4} \times n_{\text{train}}$ are smaller than or equal to the 70% quantile, while the tail of the data corresponds to test points where the ridge leverage score is larger than the 70% quantile. This stratification of the dataset allows us to visualize how the regressor performs in dense (small RLS) and sparser (large RLS) groups of the dataset. We calculate the symmetric mean absolute percentage error (SMAPE): $\frac{1}{n}\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{(|y_i|+|\hat{y}_i|)/2}$ of each group. The SMAPE is an extension of the MAPE, which overcomes the asymmetry of the latter. The results for exact sampling algorithms are visualized on Figure 2.2.1, approximate algorithms are given on Figure 2.2.2. Combining multiple subsets shows a reduction in error. Following the previous section, sampling a more diverse subset improves the performance of the KRR. Particularly diverse sampling has comparable performance for the bulk data, while performing much better in the tail of the data. Importantly, all the methods reach a stable performance before the number of points used by all interpolators exceeds the total number of training points.

2.2.4 Parameters and dataset descriptions

The parameters and datasets used in the simulations can be found in Table 2.2.1. The dataset dimensions are given by n and d, σ is the bandwidth of the Gaussian kernel, k the size of the subset. The regularization parameter of the RLS is equal to λ_{RLS}. The parameters for DPP-VFX correspond to $\bar{q}_{\text{xdpp}}$ and $\bar{q}_{\text{bless}}$.

These are the oversampling parameters for internal Nyström approximation of BLESS and DPP-VFX used to guarantee that everything terminates. Tuning parameters of the BLESS algorithm are q_0, c_0, c_1, c_2.

Table 2.2.1: Datasets and parameters used in the experiments.

Dataset	n	d	σ	k	λ_{RLS}	$\bar{\text{q}}_{\text{xdpp}}$	$\bar{\text{q}}_{\text{bless}}$	q_0	c_0	c_1	c_2
`Adult`	48842	110	5	250	10^{-3}	3	3	2	2	3	3
`Abalone`	4177	8	3	50	10^{-4}	/	/	/	/	/	/
`Wine Quality`	6497	11	5	100	10^{-4}	/	/	/	/	/	/
`Bike Sharing`	17389	16	3	250	10^{-3}	3	3	2	2	3	3
`CASP`	45730	9	2	250	10^{-3}	3	3	2	2	3	3

2.3 Conclusion

In Section 2.1, we analyzed more in-depth the relationship between diversity sampling, bias in the data, and implicit regularization. A loss was introduced to measure performance separately in the tail and bulk of the data. The bulk and tail of the data correspond to points with low and high outlyingness respectively, where the outlyingness is measured by the ridge leverage scores. By splitting the loss function into two parts, one can identify if the model is not only focusing on the majority data but also performing well in 'outlying' subpopulations. It is shown that diversity sampling performs better in these outlying regions while maintaining a comparable performance in the bulk of the data. By using DPPs for sampling, we obtained theoretical results concerning the interplay between diversity and regularization. Empirically, we demonstrated that diverse sampling methods improve performance in a number of kernel methods compared to uniform sampling. The interplay between diversity and implicit regularization was analyzed in the Nyström approximation, KPCA, and KRR. In Section 2.2, we extended the implicit regularization results to kDPPs in the context of ridgeless kernel regression. Furthermore, we leveraged the common setup of state-of-the-art DPP algorithms to sample multiple small subsets and use them in an ensemble of ridgeless regressions.

Chapter 3

Diversity sampling methods

This chapter comprises previously published and submitted articles including:

- Michaël Fanuel, Joachim Schreurs[1], and Johan AK Suykens. Nyström landmark sampling and regularized Christoffel functions. In *arXiv preprint arXiv:1905.12346*, 2019.
- Joachim Schreurs[2], Michaël Fanuel, and Johan AK Suykens. Towards deterministic diverse subset sampling. In *Proc. of Bogaerts B. et al. (eds) Artificial Intelligence and Machine Learning. Communications in Computer and Information Science, vol 1196. Springer, Cham. (BNAIC 2019, Benelearn 2019)*, pages 137-151, 2019

3.1 Nyström landmark sampling and regularized Christoffel functions

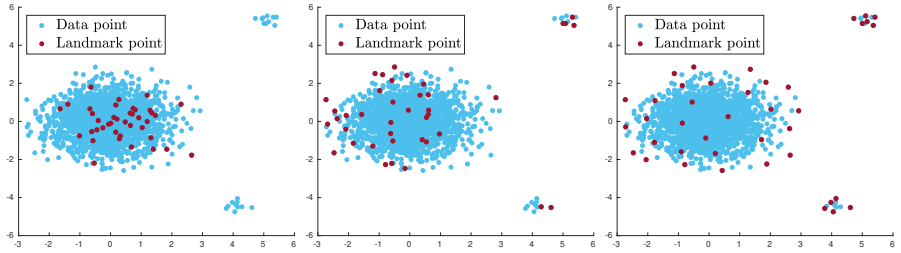
3.1.1 Introduction

This section explains how DPPs and leverage scores sampling methods can be connected in the context of Christoffel functions. These functions are known in

[1]The candidate collaborated with the first author on the research, methodology, and the article preparation, with a focus on the experiments.

[2]The candidate performed the research and wrote the article in collaboration with the coauthors.

approximation theory and defined in Appendix B.1. The Christoffel function is related to the empirical density in the sense that the function takes larger values where the data set is dense and smaller values in sparse regions. This is the main motivation for the use of the Christoffel function in this section, as they can be used to find sparse points in the data that would improve the diversity. They have been 'kernelized' in the last years as a tool for machine learning [13, 177, 129] and are also interestingly connected to the problem of support estimation [202].



(a) Uniform (b) RLS (c) DAS

Figure 3.1.1: Samplings of an artificial data set. Uniform sampling oversamples dense parts. RLS overcomes this limitation while DAS (proposed in this section) promotes more diversity.

Roadmap.

1. We start with an overview of RLS and DPP sampling for kernel methods. The connection between the two sampling methods is emphasized: both make use of the same matrix that we call here the projector kernel matrix. We also explain Christoffel functions, which are known to be connected to RLSs.

2. Motivated by this connection, we introduce conditioned Christoffel functions. These *conditioned* Christoffel functions are shown to be connected to *regularized* conditional probabilities for DPPs in Section 3.1.3, yielding a relation between Ridge Leverage Score sampling and DPP sampling.

3. Motivated by this analogy, two landmark sampling methods are analyzed: Deterministic Adaptive Selection (DAS) and Randomized Adaptive Sampling (RAS) respectively in Section 3.1.4 and Section 3.1.5. Theoretical analyses of both methods are provided for the accuracy of the Nyström approximation. In particular, RAS is a simplified and

weighted version of sequential sampling of a DPP [182, 60]. Although slower compared to non-adaptive methods, RAS achieves a better kernel approximation and produces samples with a similar diversity compared to DPP sampling. The deterministic algorithm DAS is shown to be more suited to smaller-scale data sets with a fast spectral decay.

4. Numerical results illustrating the advantages of our methods are given in Section 3.1.6. The proofs of our results are given in Appendix B.

3.1.2 Kernel matrix approximation and determinantal point processes

Let K be an $n \times n$ positive semidefinite kernel matrix. In this context, landmarks are specific data points, within a data set, which are chosen to construct a low-rank approximation of K. These points can be sampled independently by using some weights such as the ridge leverage score or thanks to a point process that accounts for both the importance and diversity of data points. These two approaches can be related to the framework of Christoffel functions.

Nyström approximation

In the context of Nyström approximations, the low rank kernel approximation relies on the selection of a subset of data points $\{\boldsymbol{x}_i | i \in \mathcal{C}\}$ for $\mathcal{C} \subseteq [n]$. This is a special case of 'optimal experiment design' [80]. For a small budget $|\mathcal{C}|$ of data points, it is essential to select points so that they are sufficiently diverse and representative of all data points in order to have the best kernel approximation possible. These points are called *landmarks* in order to emphasize that we aim to obtain both important and diverse data points. It is common to associate to a subset a sparse sampling matrix.

Definition 3.1.1 (Sampling matrix)**.** *Let $\mathcal{C} \subseteq [n]$. A sampling matrix $S(\mathcal{C})$ is an $n \times |\mathcal{C}|$ matrix obtained by selecting and possibly scaling by a strictly positive factor the columns of the identity matrix which are indexed by $\mathcal{C}$. When the rescaling factors are all ones, the corresponding unweighted sampling matrix is simply denoted by C.*

A simple connection between weighted and unweighted sampling matrices goes as follows. Let $S(\mathcal{C})$ be a sampling matrix. Then, there exists a vector of strictly positive weights $\boldsymbol{w} \in (\mathbb{R}_{>0})^n$ such that $S(\mathcal{C}) = C \operatorname{Diag}(\boldsymbol{w})$, where C is an unweighted sampling matrix. An unweighted sampling matrix can be used to select submatrices as follows: let A be an $n \times n$ matrix and $\mathcal{C} \subseteq [n]$, then

$A_{\mathcal{C}} = AC$ and $A_{\mathcal{C}\mathcal{C}} = C^\top AC$. In practice, it is customary to use the unweighted sampling matrix to calculate the kernel approximation whereas the weighted sampling matrix is convenient to obtain theoretical results.

Definition 3.1.2 (Nyström approximation)**.** *Let K be an $n \times n$ positive semidefinite symmetric matrix and let $\mathcal{C} \subseteq [n]$. Given $\mu > 0$ and an $n \times |\mathcal{C}|$ sampling matrix $S(\mathcal{C})$, the* regularized *Nyström approximation of K is*

$$L_{\mu,S}(K) = KS(S^\top KS + \mu \mathbb{I}_{|\mathcal{C}|})^{-1} S^\top K,$$

where $S = S(\mathcal{C})$. The common *Nyström approximation is simply obtained in the limit $\mu \to 0$ as follows $L_{0,S}(K) = KS(S^\top KS)^+ S^\top K$.*

The following remarks are in order.

- First, in this section, the kernel is assumed to be strictly positive definite. In this case, the common Nyström approximation involves the inverse of the kernel submatrix in the above definition rather than the pseudo-inverse.
- Second, the Nyström approximation satisfies the structural inequality

  $$0 \preceq K - L_{\mu,S}(K),$$

  for all sampling matrices S and for all $\mu \geq 0$. This can be shown thanks to Lemma B.2.3 in the Appendix.
- Third, if $S = C\,\mathrm{Diag}(\boldsymbol{w})$ with $\boldsymbol{w} \geq 1$, then we have

  $$K - L_{0,C}(K) \preceq K - L_{\mu,C}(K) \preceq K - L_{\mu,S}(K),$$

  for all $\mu > 0$, in view of Lemma B.2.3 in the Appendix. The above inequality indicates that we can bound the error on the common Nyström approximation by upper bounding the error on the regularized Nyström approximation if the weights of the sampling matrix are larger than 1.

Projector Kernel Matrix

Ridge Leverage Scores and DPPs can be used to sample landmarks. The former allows guarantees with high probability on the Nyström error. The latter allows particularly simple error bounds on expectation as we also explain hereafter. Both these approaches and the algorithms proposed in this section involve the use of another matrix constructed from K and that we call the *projector kernel matrix*, which is defined as follows

$$P_{n\gamma}(K) = K(K + n\gamma \mathbb{I})^{-1}, \tag{3.1.1}$$

where $\gamma > 0$ is a regularization parameter. This parameter is used to filter out small eigenvalues of K. When no ambiguity is possible, we will simply omit the dependence on $n\gamma$ and K to ease the reading. The terminology 'projector' can be understood as follows: if K is singular, then $P_{n\gamma}(K)$ is a mollification of the projector KK^+ onto the range of K.

We review the role of P for the approximation of K with RLS and DPP sampling in the next two subsections. These considerations motivate the algorithms proposed in this section where P plays a central role.

RLS sampling

Ridge Leverage Scores are a measure of the uniqueness of data points and are defined as follows

$$\ell_{n\gamma,K}(\boldsymbol{x}_z) = [P_{n\gamma}(K)]_{zz}, \tag{3.1.2}$$

with $z \in [n]$. When no ambiguity is possible, the dependence on K is omitted in the above definition. Leverage score sampling is used to design approximation schemes in kernel methods as, e.g., in [165] and [201]. The trace of the projector kernel matrix is the so-called effective dimension [260], i.e., $d_{\text{eff}}(\gamma) = \sum_{i\in[n]} \ell_{n\gamma}(\boldsymbol{x}_i)$. For a given regularization $\gamma > 0$, this effective dimension is typically associated with the number of landmarks needed to have an 'accurate' low rank Nyström approximation of a kernel matrix as it is explained, e.g., in [73].

To provide an intuition of the kind of guarantees obtained thanks to RLS sampling, we adapt an error bound from [32] for Nyström approximation holding with high probability, which was obtained originally in [73] in a slightly different form. To do so, we firstly define the probability to sample $i \in [n]$ by

$$p_i = \ell_{n\gamma}(\boldsymbol{x}_i)/d_{\text{eff}}(\gamma).$$

Then, RLS sampling goes as follows: one samples independently with replacement m landmarks indexed by $\mathcal{C} = \{i_1, \ldots, i_m\} \subseteq [n]$ according to the discrete probability distribution given by p_i for $1 \leq i \leq n$. Then, the associated sampling matrix is $S = C\,\mathrm{Diag}(1\sqrt{m\boldsymbol{p}})$ with $\boldsymbol{p} = (p_{i_1}, \ldots, p_{i_m})^\top$.

Proposition 3.1.1 ([73] Thm. 2 and App. A, Lem. 1)**.** *Let $\gamma > 1/n$ be the regularization parameter. Let the failure probability be $0 < \delta < 1$ and fix a parameter $0 < t < 1$. If we sample m landmarks $\mathcal{C} = \{i_1, \ldots, i_m\} \subseteq [n]$ with RLS sampling with*

$$m \geq \frac{2d_{\text{eff}}(\gamma) + 1/3}{t^2} \log\left(\frac{n}{\delta}\right)$$

and define the sampling matrix $S = C\,\mathrm{Diag}(1/\sqrt{m\boldsymbol{p}})$*, then, with probability at least* $1-\delta$*, we have*

$$K - L_{n\gamma,S}(K) \preceq \frac{n\gamma}{1-t} P_{n\gamma}(K) \preceq \frac{n\gamma}{1-t}\mathbb{I}.$$

It is interesting to notice that the bound on Nyström approximation is improved if γ decreases, which also increases the effective dimension of the problem. To fix the ideas, it is common to take $t = 1/2$ in this type of result as it is done in [73] and [165] for symmetry reasons within the proofs.

Remark 3.1.1 (The role of column sampling and the projector kernel matrix)**.** *The basic idea for proving Proposition 3.1.1 is to decompose the kernel matrix as* $K = B^\top B$*. Then, we have* $P_{n\gamma}(K) = \Psi^\top\Psi$ *with* $\Psi = (BB^\top + n\gamma\mathbb{I})^{-1/2}B$ *as it can be shown by the push-through identity given in Lemma B.2.2 in the Appendix. Next, we rely on an independent sampling of columns of* Ψ *which yields* ΨS*, where* S *is a weighted sampling matrix. Subsequently, a matrix concentration inequality, involving a sum of columns sampled from* Ψ*, is used to provide an upper bound on the largest eigenvalue of* $\Psi\Psi^\top - \Psi SS^\top\Psi^\top$*. Finally, the following key identity*

$$K - L_{n\gamma,S}(K) = n\gamma\Psi^\top\left(\Psi SS^\top\Psi^\top + n\gamma(BB^\top + n\gamma\mathbb{I})^{-1}\right)^{-1}\Psi,$$

is used to connect this bound with Nyström approximation. A similar strategy is used within this section to yield guarantees for RAS hereafter.

DPP sampling

Let us discuss the use of DPP sampling for Nyström approximation. We define below a class of DPPs, called L-ensembles, by following [106] and [125].

Definition 3.1.3 (L-ensemble)**.** *A finite Determinantal Point Process* $\mathbf{Y}$ *is an L-ensemble if the probability to sample the set* $\mathcal{C} \subseteq [n]$ *is given by*

$$\Pr(\mathbf{Y} = \mathcal{C}) = \frac{\det(L_{\mathcal{C}\mathcal{C}})}{\det(\mathbb{I} + L)}$$

where L *is an* $n \times n$ *matrix whose principal minors are all non-negative. The corresponding process is denoted here by* $DPP_L(L)$*.*

In the above definition, the normalization factor is obtained by requiring that the probabilities sum up to unity. Importantly, for the specific family of matrices

$$L = K/(n\gamma), \text{ such that } K \succeq 0 \text{ is a kernel matrix and } \gamma > 0,$$

the marginal inclusion probabilities of this process $\Pr(\mathcal{C} \subseteq \mathbf{Y}) = \det(P_{\mathcal{C}\mathcal{C}})$ are associated with the following marginal kernel $P = P_{n\gamma}(K)$, which is defined in (3.1.1).

Sequential DPP sampling

Let $\mathcal{C}$ and $\tilde{\mathcal{C}}$ be disjoint subsets of $[n]$. The general conditional probability of a DPP both involves a condition of the type $\mathcal{C} \subseteq \mathbf{Y}$ and $\mathbf{Y} \cap \tilde{\mathcal{C}} = \emptyset$. The formula for the general conditional probability of a DPP is particularly useful for sequential sampling, which is also related to the sampling algorithms proposed in this section.

It was noted recently in [182] and [60] that DPP sampling can be done in a sequential way. We consider here the case of an L-ensemble DPP. The sequential sampling of a DPP involves a sequence of n Bernouilli trials. Each Bernouilli outome determines if one item is accepted or rejected. The algorithm loops over the set $[n]$ and constructs sequentially the disjoint sets $\mathcal{C}$ and $\tilde{\mathcal{C}}$ which are initially empty. At any step, the set $\mathcal{C}$ contains accepted items while $\tilde{\mathcal{C}}$ consists of items that were rejected. At step i, the element $i \in [n]$ is sampled if the outcome of a $\mathrm{Bern}(p_i)$ random variable is successful. The success probability is defined as

$$p_i = \Pr(i \in \mathbf{Y} | \mathcal{C} \subseteq \mathbf{Y}, \mathbf{Y} \cap \tilde{\mathcal{C}} = \emptyset),$$

where $\mathbf{Y} \sim DPP_L(L)$. Then, if i is sampled successfully, we define $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$ and otherwise $\tilde{\mathcal{C}} \leftarrow \tilde{\mathcal{C}} \cup \{i\}$. The general conditional probability is given by

$$\Pr(z \in \mathbf{Y} | \mathcal{C} \subseteq \mathbf{Y}, \mathbf{Y} \cap \tilde{\mathcal{C}} = \emptyset) = [H^{\tilde{\mathcal{C}}} - L_{0,C}(H^{\tilde{\mathcal{C}}})]_{zz}, \tag{3.1.3}$$

where $\mathcal{C} \cap \tilde{\mathcal{C}} = \emptyset$ and the 'conditional marginal kernel' is

$$H^{\tilde{\mathcal{C}}} = P + P_{\tilde{C}} \big(\mathbb{I} - P_{\tilde{C}\tilde{C}}\big)^{-1} P_{\tilde{C}}^{\top}, \tag{3.1.4}$$

with $P = L(L+\mathbb{I})^{-1}$. The expression (3.1.3) can be found in a slightly different form in [60, Corollary 1]. From a computational perspective, P (or $\mathbb{I} - P$) may still have numerically very small eigenvalues so that the inverse appearing in the calculation of the Nyström approximation (3.1.8) or in (3.1.4) has to be regularized.

An exact sampling of a discrete DPP of a set of n items requires $O(n^3)$ operations. It is therefore interesting to analyze different sampling methods which can also yield diverse sampling since exact DPP sampling is often prohibitive for large n. The role of DPP sampling for approximating kernel matrices motivates the next section where we extend the analogy between Christoffel functions and general conditional probabilities of L-ensemble DPPs.

3.1.3 Christoffel functions and determinantal point processes

Kernelized Christoffel functions

In the case of an empirical density, the value of the regularized Christoffel function at $x \in \mathbb{R}^d$ is obtained as follows:

$$C_{\boldsymbol{\eta}}(x) = \inf_{f \in \mathcal{H}} \sum_{i=1}^{n} \frac{\eta_i}{n} f(\boldsymbol{x}_i)^2 + \gamma \|f\|_{\mathcal{H}}^2 \text{ s.t. } f(\boldsymbol{x}) = 1, \tag{CF}$$

where $\eta_i \geq 0$ for $i \in [n]$ are pre-determined weights that can be used to give more or less emphasis to data points. A specific choice of weights is studied in Section 3.1.3. Notice that, in the above definition, there is an implicit dependence on the parameter $\gamma > 0$, the empirical density, and the RKHS itself. The functions defined by (CF) are related to the empirical density in the sense that $C_{\boldsymbol{\eta}}(x)$ takes larger values where the data set is dense and smaller values in sparse regions. This is the intuitive reason why Christoffel functions can be used for outlier detection [13]. As it is explained in [177, Eq. (2)], the Christoffel function takes the following value on the data set

$$C_{\boldsymbol{\eta}}(\boldsymbol{x}_i) = \left(\boldsymbol{e}_i^\top K \Big(K \operatorname{Diag}\left(\frac{\boldsymbol{\eta}}{n}\right) K + \gamma K \Big)^{-1} K \boldsymbol{e}_i \right)^{-1} \text{ for all } i \in [n]. \tag{3.1.5}$$

The function defined by (CF) naturally provides importance scores of data points. The connection with RLSs (see (3.1.1) and (3.1.2)) when $\boldsymbol{\eta} = 1$ can be seen thanks to the equivalent expression

$$C_{\boldsymbol{\eta}}(\boldsymbol{x}_i) = \frac{\eta_i}{n} \left(\boldsymbol{e}_i^\top K \left(K + n\gamma \operatorname{Diag}(\boldsymbol{\eta})^{-1} \right)^{-1} \boldsymbol{e}_i \right)^{-1} \text{ for all } i \in [n], \tag{3.1.6}$$

which is readily obtained thanks to the matrix inversion lemma. For many data sets, the 'importance' of points is not uniform over the data set, which motivates the use of RLS sampling and our proposed methods as it is illustrated in Figure 3.1.1. For completeness, we give a derivation of (3.1.5) in Appendix B.1. Clearly, if $\boldsymbol{\eta} > 0$, this Christoffel function can also be understood as a RLS associated to a reweighted kernel matrix $\bar{K}(\boldsymbol{\eta}) = \operatorname{Diag}(\boldsymbol{\eta})^{-1/2} K \operatorname{Diag}(\boldsymbol{\eta})^{-1/2}$, i.e.,

$$C_{\boldsymbol{\eta}}(\boldsymbol{x}_i) = n^{-1} / [\bar{K}(\boldsymbol{\eta}) \left(\bar{K}(\boldsymbol{\eta}) + n\gamma \mathbb{I} \right)^{-1}]_{ii} \tag{3.1.7}$$

for all $i \in [n]$.

We first define conditioned Christoffel functions and show their connection with a conditional probability for an L-ensemble DPP. This helps to provide some intuition and is a warm-up for the more general results thereafter.

Conditioned Christoffel functions

We extend (CF) in order to provide importance scores of data points which are in the complement $\mathcal{C}^c$ of a set $\mathcal{C} \subseteq [n]$. This consists in excluding points of $\mathcal{C}$ – and, implicitly, to some extend also their neighbourhood – in order to look for other important points in $\mathcal{C}^c$. Namely, we define a conditioned Christoffel function as follows:

$$C_{\mathcal{C},\boldsymbol{\eta}}(\boldsymbol{x}_z) = \inf_{f \in \mathcal{H}} \sum_{i=1}^{n} \frac{\eta_i}{n} f(\boldsymbol{x}_i)^2 + \gamma \|f\|_{\mathcal{H}}^2 \text{ s.t. } \begin{cases} f(\boldsymbol{x}_z) = 1 \text{ and,} \\ f(\boldsymbol{x}_s) = 0 \ \forall s \in \mathcal{C}. \end{cases} \tag{CondCF}$$

where we restrict here to $\boldsymbol{x}_z \in X$ such that $z \in \mathcal{C}^c$. In order to draw a connection with leverage scores, we firstly take uniform weights $\eta_i = 1$ for all $i \in \mathcal{C}^c$. In this section, we propose to solve several problems of the type (CondCF) for different sets $\mathcal{C} \subseteq [n]$. Then, we will select landmarks x_z which minimize $C_{\mathcal{C},\eta}(x_z)$ in the data set for a certain choice of η.

Remark 3.1.2 (Extended weights)**.** *Non-uniform weights are considered below in Proposition 3.1.4. We anticipate that the constraint $f(x_s) = 0$ for all $s \in \mathcal{C}$ can be implemented by considering $\eta_s = +\infty$ for all $s \in \mathcal{C}$, in the spirit of an indicator function of a set as it is commonly used in mathematical optimization.*

Now, we give a connection between Nyström approximation and conditioned Christoffel functions. Hereafter, Proposition 3.1.2 gives a closed expression for the conditioned Christoffel function in terms of the common Nyström approximation of the projector kernel matrix. Before stating this result, we introduce the following submatrices of the projector kernel: $P_{\mathcal{C}} = PC$ and $P_{\mathcal{C}\mathcal{C}} = C^\top PC$, where $C \in \mathbb{R}^{n \times |\mathcal{C}|}$ is an unweighted sampling matrix.

Proposition 3.1.2. *Let k be a strictly positive definite kernel and let $\mathcal{H}$ be the associated RKHS. For $\mathcal{C} \subset [n]$, we have*

$$C_{\mathcal{C},1}(\boldsymbol{x}_z) = n^{-1} / [P - P_{\mathcal{C}} P_{\mathcal{C}\mathcal{C}}^{-1} P_{\mathcal{C}}^\top]_{zz}, \tag{3.1.8}$$

for all $z \in \mathcal{C}^c$.

Clearly, in the absence of conditioning, i.e., when $\mathcal{C} = \emptyset$, one recovers the result of [177] which has related the Christoffel function to the inverse leverage score. For $\mathcal{C} \neq \emptyset$, the second term at the denominator of (3.1.8) is clearly the common Nyström approximation (see Definition 3.1.2) of the projector kernel based on the subset of columns indexed by $\mathcal{C}$. Also, notice that $P - P_{\mathcal{C}} P_{\mathcal{C}\mathcal{C}}^{-1} P_{\mathcal{C}}^\top$ is the posterior predictive variance of a Gaussian process regression associated with the projector kernel P in the absence of regularization [186]. The accuracy of

the Nyström approximation based on $\mathcal{C}$ is given in terms of the max norm as follows:

$$\min_{z \in \mathcal{C}^c} C_{\mathcal{C},1}(\boldsymbol{x}_z) = n^{-1}/\|P - L_{0,\mathcal{C}}(P)\|_\infty,$$

where $L_{0,C}$ is the common low-rank Nyström approximation.

Now, we derive a connection between conditioned Christoffel functions and conditional probabilities for a specific DPP: an L-ensemble which is built thanks to a rescaled kernel matrix. To begin, we provide a result relating conditioned Christoffel functions and determinants in the simple setting of the last section. Below, Proposition 3.1.3, which can also be found in [18] in a slightly different setting, gives an equivalent expression for a conditioned Christoffel function.

Proposition 3.1.3 (Determinant formula)**.** *Let $\mathcal{C} \subseteq [n]$ a non-empty set and $z \in \mathcal{C}^c$. Denote $\mathcal{C}_z = \mathcal{C} \cup \{z\}$. For $\boldsymbol{\eta} = 1$, the solution of* (CondCF) *has the form:*

$$C_{\mathcal{C},1}(\boldsymbol{x}_z) = n^{-1} \frac{\det P_{\mathcal{C}\mathcal{C}}}{\det P_{\mathcal{C}_z\mathcal{C}_z}}. \tag{3.1.9}$$

Furthermore, let B a matrix such that $P = B^\top B$. Then, we have the alternative expression $C_{\mathcal{C},1}(\boldsymbol{x}_z) = n^{-1}/\|\boldsymbol{b}_z - \pi_{V_\mathcal{C}} \boldsymbol{b}_z\|_2^2$, where $\boldsymbol{b}_z \in \mathbb{R}^{N \times 1}$ is the z-th column of B and $\pi_{V_\mathcal{C}}$ is the projector on $V_\mathcal{C} = \operatorname{span}\{b_s | s \in \mathcal{C}\}$.

Interestingly, the determinants in (3.1.9) can be interpreted in terms of probabilities associated to an L-ensemble DPP with $L = K/(n\gamma)$. In other words, Proposition 3.1.3 states that the conditioned Christoffel function is inversely proportional to the conditional probability

$$\Pr(z \in \mathbf{Y} | \mathcal{C} \subseteq \mathbf{Y}) = \frac{\det P_{\mathcal{C}_z\mathcal{C}_z}}{\det P_{\mathcal{C}\mathcal{C}}}, \tag{3.1.10}$$

for $\mathbf{Y} \sim DPP_L(K/(n\gamma))$. Hence, minimizing $C_{\mathcal{C},1}(\boldsymbol{x}_z)$ over $z \in [n]$ produces an optimal $z^\star$ which is diverse with respect to $\mathcal{C}$. Inspired by the identity (3.1.10), we now provide a generalization in Section 3.1.3, which associates a common Christoffel function to a general conditional probability of a DPP by using a specific choice of weights $\boldsymbol{\eta}$.

Regularized conditional probabilities of a DPP and Christoffel functions

Interestingly, the regularization of the expression of the general conditional probabilities for a DPP in (3.1.3) can be obtained by choosing specific weights $\boldsymbol{\eta}$ in the definition of the Christoffel function. To do so, the hard constraints in (CondCF) are changed to soft constraints in Proposition 3.1.4; see Figure 3.1.2 for an illustration. In other words, the value of $f \in \mathcal{H}$ is severely penalized on

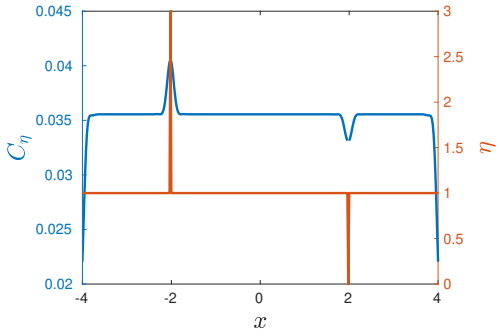


Figure 3.1.2: Christoffel function $C_{\boldsymbol{\eta}(\mu,\nu)}(\boldsymbol{x}_i)$ (blue line); see Proposition 3.1.4 for the precise definition. We take $\gamma = 0.01$ and the Gaussian kernel's bandwidth is $\sigma = 0.2/\sqrt{2}$. The data set is the following grid $\boldsymbol{x}_i = -4 + 0.02i$ for all the integers $0 \leq i \leq 400$. The weights $\eta_i(\mu,\nu)$ for $\mu = 0.5$ and $\nu = 0.01$ are displayed in red whereas the corresponding sets are $\mathcal{C} = \{100\}$ and $\tilde{\mathcal{C}} = \{300\}$.

$\boldsymbol{x}_i$ when $i \in \mathcal{C}$ in the optimization problem given in (CondCF). In the same way, f can also be weakly penalized on another disjoint subset $\tilde{\mathcal{C}}$.

Proposition 3.1.4 (Conditioning by weighting)**.** *Let $\mu > 0$ and $0 < \nu < 1$. Let two disjoint subsets $\mathcal{C}, \tilde{\mathcal{C}} \subset [n]$ and define C and $\tilde{C}$ the matrices obtained by selecting the columns of the identity indexed by $\mathcal{C}$ and $\tilde{\mathcal{C}}$ respectively. Let $P = P_{n\gamma}(K)$ and define the following weights:*

$$\eta_i(\mu,\nu) = \begin{cases} 1+\mu^{-1}, & \text{if } i \in \mathcal{C}, \\ \nu, & \text{if } i \in \tilde{\mathcal{C}}, \\ 1, & \text{otherwise}, \end{cases}.$$

Then, the Christoffel function defined by (CF) *associated to $\boldsymbol{\eta}(\mu,\nu)$ is given by*

$$C_{\boldsymbol{\eta}(\mu,\nu)}(\boldsymbol{x}_z) = n^{-1} / \left[H_\nu^{\tilde{\mathcal{C}}} - L_{\mu,C}\left(H_\nu^{\tilde{\mathcal{C}}}\right) \right]_{zz},$$

where the regularized 'conditional marginal kernel' is

$$H_\nu^{\tilde{\mathcal{C}}} = P + (1-\nu) P_{\tilde{C}} \left(\mathbb{I} - (1-\nu) P_{\tilde{C}\tilde{C}} \right)^{-1} P_{\tilde{C}}^\top.$$

Hence, Proposition 3.1.4 shows that by choosing the weights $\boldsymbol{\eta}$ appropriately the corresponding Christoffel function gives a regularized formulation of conditional probabilities of a DPP if $\mu > 0$ and $0 < \nu < 1$ are small enough. Explicitly, we have

$$\lim_{\mu \to 0} \lim_{\nu \to 0} C_{\boldsymbol{\eta}(\mu,\nu)}(\boldsymbol{x}_z) = n^{-1} / \Pr(z \in \mathbf{Y} | \mathcal{C} \subseteq \mathbf{Y}, \mathbf{Y} \cap \tilde{\mathcal{C}} = \emptyset),$$

for $\mathbf{Y} \sim DPP_L(K/(n\gamma))$ and $z \in [n]$, while the conditional probability on the RHS in given in (3.1.3). It is instructive to view the above Christoffel function as a Ridge Leverage Score of a reweighted matrix, bearing in mind the expression of (3.1.7),

$$C_{\boldsymbol{\eta}(\mu,\nu)}(\boldsymbol{x}_i) = n^{-1}\left[\bar{K}(\boldsymbol{\eta})\left(\bar{K}(\boldsymbol{\eta}) + n\gamma\mathbb{I}\right)^{-1}\right]^{-1}_{ii},$$

for all $i \in [n]$ and where the reweighted kernel matrix is $[\bar{K}(\boldsymbol{\eta})]_{ij} = \frac{K}{\sqrt{\eta_i \eta_j}}$ for all $1 \leq i, j \leq n$. We emphasize that if μ and ν take small numerical values, the rows and columns of the reweighted matrix $\bar{K}$ indexed by a subset $\mathcal{C}$ have now very small values while the rows and columns indexed by $\tilde{\mathcal{C}}$ have large numerical values.

3.1.4 Deterministic adaptive landmark selection

It is often difficult to obtain theoretical guarantees for deterministic landmark selection. We give here an approach for which some guarantees can be given. Inspired by the connection between DPPs and Christoffel functions, we propose to select deterministically landmarks by finding the minima of conditioned Christoffel functions defined such that the additional constraints enforce diversity. As it is described in Algorithm 3.1.1, we can successively minimize $C_{\mathcal{C}}(\boldsymbol{x}_z)$ over a nested sequence of sets $\mathcal{C}_0 \subseteq \mathcal{C}_1 \subseteq \cdots \subseteq \mathcal{C}_m$ starting with $\mathcal{C}_0 = \emptyset$ by adding one landmark at each iteration. It is not excluded that the discrete maximization problem in Algorithm 3.1.1 can have several maxima. In practice, this is rarely the case. The ties can be broken uniformly at random or by using any deterministic rule. Hence, in the former case, the algorithm is not deterministic anymore. In fact, Algorithm 3.1.1 is a greedy reduced basis method as it is defined in [61], which has been used recently for landmarking on manifolds in [80] with a different kernel. Its convergence is studied in Proposition 3.1.5.

Proposition 3.1.5 (Convergence of DAS)**.** *Let* $\Lambda_1 \geq \ldots \Lambda_n > 0$ *be the eigenvalues of the projector kernel matrix* P *defined in* (3.1.1)*. If* $\mathcal{C}_m$ *is sampled*

Algorithm 3.1.1 DAS.

input: Matrix $K \succ 0$, sample size k and $\gamma > 0$.
initialization: $\mathcal{C}_0 = \emptyset$ and $m = 1$.
while: $m \leq k$ **do**
  $s_m \in \arg\max \mathrm{Diag}\left(P - P_{\mathcal{C}_m} P^{-1}_{\mathcal{C}_m \mathcal{C}_m} P^{\top}_{\mathcal{C}_m}\right)$.
  $\mathcal{C}_m \leftarrow \mathcal{C}_{m-1} \cup \{s_m\}$ and $m \leftarrow m + 1$.
end while
return $\mathcal{C}_m$.

according to Algorithm 3.1.1, we have:

$$\|P - P_{\mathcal{C}_m} P_{\mathcal{C}_m \mathcal{C}_m}^{-1} P_{\mathcal{C}_m}^{\top}\|_{\infty} \leq 2\|P\|_{\infty} \Lambda_{\lfloor m/2 \rfloor + 1}^{1/2}, \text{ for all } 2 \leq m < n.$$

Notice that $\|P\|_{\infty}$ is indeed the largest leverage score related to the maximal marginal degrees of freedom defined in [15]. Furthermore, we remark that the eigenvalues of P are related to the eigenvalues $\lambda_1 \geq \ldots \lambda_n > 0$ of the kernel matrix K by the Tikhonov filtering $\Lambda_m = \frac{\lambda_m}{\lambda_m + n\gamma}$. In the simulations, we observe that DAS performs well when the spectral decay is fast. This seems to be often the case in practice as explained in [176] and [155]. Hence, since DPP sampling is efficient for obtaining an accurate Nyström approximation, this deterministic adaptation can be expected to produce satisfactory results. However, it is more challenging to obtain theoretical guarantees for the accuracy of the Nyström approximation obtained from the selected subset. For a set of landmarks, $\mathcal{C}$, Lemma 3.1.1 relates the error on the approximation of K to the error on the approximation of P.

Lemma 3.1.1. *Let $\mu > 0$ and $\tilde{\mu} = \mu/(\lambda_{max}(K) + n\gamma)$. Then, it holds that:*

$$0 \preceq \frac{K - L_{\mu,S}(K)}{\lambda_{max}(K) + n\gamma} \preceq P_{n\gamma}(K) - L_{\tilde{\mu},S}(P_{n\gamma}(K)).$$

This structural result connects the approximation of K to the approximation of P as a function of the regularization $n\gamma$. However, the upper bound on the approximation of K can be loose since $\lambda_{max}(K)$ can be large in general. It is also true that we might expect the Nyström approximation error to go to zero when the number of landmarks goes to n. Nevertheless, the rate of convergence to zero is usually unknown for deterministic landmark selection methods. For more instructive theoretical guarantees, we study a randomized version of Algorithm 3.1.1 in the next section.

Remark 3.1.3 (Related approaches)**.** *Several successful approaches for landmarking are based on Gaussian processes [139] or use a Bayesian approach [174] in the context of active learning. These methods are similar to DAS with the major difference that DAS uses the projector kernel $P = K(K + n\gamma\mathbb{I})^{-1}$, while the above-mentioned methods rely on a kernel function $k(\boldsymbol{x}, \boldsymbol{y})$ or its corresponding kernel matrix.*

3.1.5 Randomized adaptive landmark sampling

Ridge Leverage Score sampling and L-ensemble DPP sampling both allow for obtaining guarantees for kernel approximation as it is explained in Section 3.1.2.

Randomized Adaptive Sampling (RAS) can be seen as an intermediate approach that gives a high probability guarantee for Nyström approximation error with a reduced computational cost compared with DPP sampling. RAS is best understood by considering the analogy with sequential DPP sampling and Christoffel functions.

As it is mentioned in Section 3.1.3, the sequential sampling of a DPP sequentially samples items conditionally to the knowledge of previously accepted and rejected items, i.e., the disjoint sets $\mathcal{C}$ and $\tilde{\mathcal{C}}$ defined in Section 3.1.3. The computation of the matrix H in (3.1.4) involves solving a linear system to account for past rejected samples. RAS uses a similar approach with the difference that the knowledge of the past rejected items is disregarded. Another difference with respect to DPP sequential sampling is that the sampling matrix used for constructing the regularized Nyström approximation is weighted contrary to the analogous formula in (3.1.3). As explained in Algorithm 3.1.2, RAS constructs a sampling matrix by adaptively choosing to add (or not) a column to the sampling matrix obtained at the previous step.

Algorithm 3.1.2 RAS.

input: Matrix $K \succ 0$, oversampling $c > 0$ and regularizers $\gamma > 0$, $\epsilon \in (0, 1)$ and $t = 1/2$.
initialization: $S_0 = \emptyset$.
for i=1,. . . , n **do**
  calculate $s_i = \frac{1}{\epsilon}\left[P_{n\gamma}(K) - L_{\epsilon,S_{i-1}}(P_{n\gamma}(K))\right]_{ii}$.
  calculate $p_i = \min\{1, c\tilde{l}_i\}$ with $\tilde{l}_i = \min\{1, (1+t)s_i\}$.
  draw $\boldsymbol{x}_i \sim \text{Bernoulli}(p_i)$.
  if $\boldsymbol{x}_i = 1$,
    add a column to the sampling matrix as follows $S_i \leftarrow (S_{i-1}|\frac{\boldsymbol{e}_i}{\sqrt{p_i}})$.
  else
    $S_i \leftarrow S_{i-1}$ (no update).
  end if
end for
return S_n.

We now explain the sampling algorithm which goes sequentially over all $i \in [n]$. Initialize $\mathcal{S}_0 = \emptyset$. Let $i \in [n]$ and denote by S_{i-1} a sampling matrix associated to the set $\mathcal{S}_{i-1}$ at iteration $i-1$. A Bernouilli random variable determines if i is added to the landmarks or not. If Bernouilli(p_i) is successful, we update of the sampling matrix $S_i = (S_{i-1}|\frac{\boldsymbol{e}_i}{\sqrt{p_i}})$, otherwise $S_i = S_{i-1}$. The success probability

p_i is defined to depend on the score

$$s_i = \frac{1}{\epsilon}\left[P_{n\gamma}(K) - L_{\epsilon,S_{i-1}}(P_{n\gamma}(K))\right]_{ii},$$

by analogy with the conditional probability (3.1.10) and the Christoffel function value (3.1.8). The sampling probability is defined as

$$p_i = \min\{1, c\tilde{l}_i\} \quad \text{with} \quad \tilde{l}_i = \min\{1, (1+t)s_i\}, \tag{3.1.11}$$

with $t > 0$ and $c > 0$.

Remark 3.1.4 (Oversampling parameter)**.** *In this section, the positive parameter c is considered as an oversampling factor and is chosen such that $c > 1$. This lower bound is less restrictive than the condition on c in Theorem 3.1.1. In our simulations, the oversampling parameter is always strictly larger than* 1 *so that the expression of the sampling probability simplifies to $p_i = \min\{1, c(1+t)s_i\}$.*

Interpretation of the sampling probability. Here, $t > 0$ is a parameter that will take a fixed value and plays the same role as the parameter t in Proposition 3.1.1. Therefore t should not take a large value. Then, $\tilde{l}_i$ can be viewed as a leverage score. To allow for varying the number of sampled points, this leverage score is multiplied by an oversampling factor $c > 0$ as it is also done for recursive RLS sampling in [165]. The resulting score can be larger than 1 and therefore it is capped at 1. The idea is that the score s_i is an approximation of an RLS. Indeed, if we define the following factorization

$$B^\top B = P_{n\gamma}(K), \tag{3.1.12}$$

an alternative expression of the score is given by

$$s_i = [B^\top (BS_{i-1}S_{i-1}^\top B^\top + \epsilon\mathbb{I})^{-1}B]_{ii}$$

as a consequence of Lemma B.2.3 in the Appendix. Then, the score s_i is an approximation of the ridge leverage score

$$[B^\top (BB^\top + \epsilon\mathbb{I})^{-1}B]_{ii} = [P_\epsilon(P_{n\gamma}(K))]_{ii},$$

with regularization parameter $\epsilon > 0$, as it can be shown thanks to the push-through identity[1]. In view of Section 3.1.2, the above ridge leverage score can be used for sampling a subset to compute a Nyström approximation of $P_{n\gamma}(K)$ thanks to Proposition 3.1.1. However, we are interested in the Nyström approximation of K. The following identity allows us to relate s_i to the approximation of K.

[1]See Lemma B.2.2 in the Appendix.

Lemma 3.1.2. *Let $K \succ 0$. Then, we have $P_\epsilon(P_{n\gamma}(K)) = \frac{1}{1+\epsilon} P_{\frac{\epsilon n\gamma}{1+\epsilon}}(K)$.*

The second regularization parameter ϵ is typically chosen as $\epsilon = 10^{-10}$ so that, in this circumstance, the adaptivity in Algorithm 3.1.2 promotes diversity among the landmarks in view of the discussion in Section 3.1.3.

Remark 3.1.5 (Subset size)**.** *Clearly, RAS outputs a subset of random size. This is also the case for L-ensemble DPP sampling, with the notable difference that the mean and variance of the subset size are known exactly in the latter case. For RAS, we empirically observe that the size variance is not large. The parameter $\gamma > 0$ influences the effective dimension of the problem and therefore helps to vary the number of points sampled by RAS. A small γ yields a large effective dimension and conversely. Increasing c also increases the sample size which might also reduce diversity.*

Theoretical guarantees. An error bound for Nyström approximation with RAS can be obtained by analogy with RLS sampling. Our main result in Theorem 3.1.1 deals with the adaptivity of RAS to provide an analogue of Proposition 3.1.1 which was designed for independent sampling.

We can now explain how RAS can be used to obtain a Nyström approximation of K by introducing an appropriate column sampling strategy. We define the factorization

$$P_\epsilon(P_{n\gamma}(K)) = \Psi^\top \Psi,$$

with $\Psi = (BB^\top + \epsilon \mathbb{I})^{-1/2} B$ and with B defined in (3.1.12). Therefore, thanks to Lemma 3.1.3 below, we know that if we can sample appropriately the columns of Ψ, denoted by $(\boldsymbol{\psi}_i)_{i=1}^n$, then the error on the corresponding Nyström approximation will be bounded by a small constant.

Lemma 3.1.3 (Nyström approximation from column subset selection)**.** *Let $\epsilon > 0$ and $0 < t < (1+\epsilon)^{-1}$. Denote by Ψ a matrix such that $\Psi^\top \Psi = P_\epsilon(P_{n\gamma}(K))$. If there is a sampling matrix S such that*

$$\lambda_{max}(\Psi\Psi^\top - \Psi S S^\top \Psi^\top) \leq t,$$

then, it holds that

$$0 \preceq K - L_{\frac{\epsilon n\gamma}{1+\epsilon}, S}(K) \preceq \frac{\epsilon n\gamma}{1 - t(1+\epsilon)} P_\epsilon(P_{n\gamma}(K)).$$

Some remarks are in order. Firstly, Lemma 3.1.3 has been adapted from [73] in order to match our setting and its proof is given in Appendix B.4 for completeness. Secondly, the parameter t does not play a dominant role in the

above error bound as long as it is not close to $(1+\epsilon)^{-1}$. Indeed, the error bound can be simply improved by choosing a smaller value for the product $\epsilon n\gamma$. Hence, if we require $\epsilon < 1$, then we can choose $t = 1/2$ to clarify the result statement. Then, we have the simplified error bound

$$0 \preceq K - L_{\frac{\epsilon n\gamma}{1+\epsilon},S}(K) \preceq \frac{2\epsilon n\gamma}{1-\epsilon}\mathbb{I},$$

where we used that $P_\epsilon(P_{n\gamma}(K)) \preceq \mathbb{I}$ in Lemma 3.1.3.

In order to use this result, we want to find a good subset of columns of Ψ, i.e., a set of landmarks $\mathcal{S}$ and probabilities $(p_j)_{j\in\mathcal{S}}$ such that

$$\lambda_{max}\left(\sum_{i=1}^{n}\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top - \sum_{j\in\mathcal{S}}\boldsymbol{\psi}_j\boldsymbol{\psi}_j^\top/p_j\right) \leq t,$$

for $t > 0$, with high probability. This condition can be achieved for well-chosen probabilities $\{p_j : j \in \mathcal{S}\}$ by using a martingale concentration inequality which helps to deal with the adaptivity of RAS. This matrix martingale inequality plays the role of the matrix Bernstein inequality used in the proof of Proposition 3.1.1 in [73].

Below, Theorem 3.1.1 indicates that, if the oversampling c is large enough, RAS produces a sampling matrix allowing for a 'good' Nyström approximation of K with high probability. For convenience, we recall the definition of the negative branch of the Lambert function $W_{-1} : [-e^{-1}, 0) \to \mathbb{R}$ which satisfies $\lim_{y\to 0_+} W_{-1}(-y)/\log(y) = 1$. Also, define the function $g(a) = -W_{-1}(-1/a)$ for all $a \geq e$, which has the asymptotic behaviour $g(a) \sim \log(a)$.

Theorem 3.1.1 (Main result)**.** *Let $\epsilon \in (0,1)$ and $\gamma > 0$. If the oversampling parameter satisfies:*

$$c \geq \frac{28}{3}g\left(\frac{700 d_{\text{eff}}(\frac{\epsilon n\gamma}{1+\epsilon})}{3(1+\epsilon)\delta}\right) \vee \frac{1+\sqrt{37}}{3},$$

then, with a probability at least $1-\delta$, Algorithm 3.1.2 yields a weighted sampling matrix S such that

$$\|K - L_{\frac{\epsilon n\gamma}{1+\epsilon},S}(K)\|_2 \leq \frac{2\epsilon n\gamma}{1-\epsilon}.$$

Remark 3.1.6 (Common Nyström approximation)**.** *In view of the remarks of Section 3.1.2, the result of Theorem 3.1.1 directly implies a bound on the common Nyström approximation, that is $\|K - L_{0,S}(K)\|_2 \leq \frac{2\epsilon n\gamma}{1-\epsilon}$.*

Discussion. Let us list a few comments about Theorem 3.1.1. The proof of Theorem 3.1.1, given in Appendix B.4, uses the martingale-based techniques developed in [47] in the context of online sampling, together with a refinement by [161] of the Freedman-type inequality for matrix martingales due to [234]. An advantage of the Freedman inequality obtained in [161] is that it does not directly depend on the matrix dimensions but rather on a certain form of 'intrinsic' dimension. The martingale-based techniques allow to deal with the adaptivity of our algorithm which precludes the use of statistical results using independent sampling. From the theoretical perspective, the adaptivity does not improve the type of guarantees obtained for independent sampling. The adaptivity of RAS rather makes the theoretical analysis more involved. However, our simulations indicate an empirical improvement. We emphasize that the dependence on the effective dimension in Theorem 3.1.1 does not appear in [47] although the proof technique is essentially the same. This extra dependence on $d_{\text{eff}}(\frac{\epsilon n\gamma}{1+\epsilon})$ helps to account for the kernel structure although the effective dimension can be however rather large in practice if $\epsilon \approx 10^{-10}$. A small enough ϵ indeed promotes the diversity of sampled landmarks. The bound of Theorem 3.1.1 is pessimistic since it depends on the effective dimension which does not account for the subsample diversity. In its current state, the theory cannot yield improved guarantees over independent sampling. In practice, the chosen value for the oversampling factor is taken much smaller. A major difference with [73] and [165] is that the landmarks $\mathcal{S}$ are *not* sampled independently in Algorithm 3.1.2. Algorithmically, RAS also corresponds to running Kernel Online Row Sampling [33] with the difference that it is applied here on the projector kernel $P_{n\gamma}(K)$. As it is mentioned hereabove, a natural link can be made between RAS and sequential sampling of a DPP. RAS involves a regularized score $[P - L_{\epsilon,S_{i-1}}(P)]_{ii}$ at step i, where S_{i-1} is a sampling matrix associated to a set $\mathcal{C}_{i-1} \subset [n]$. Indeed, this score depends on the points $\mathcal{C}_{i-1}$ which were chosen before step i, but not on the points $\tilde{\mathcal{C}}_{i-1}$ that were *not* chosen (corresponding to a failed Bernoulli). Sequential sampling of a DPP at step i also involves drawing a Bernoulli with probability $\Pr(i \in \mathbf{Y} | \mathcal{C}_{i-1} \subseteq \mathbf{Y}, \mathbf{Y} \cap \tilde{\mathcal{C}}_{i-1} = \emptyset)$ as given by a conditional kernel in (3.1.4). The first major difference with sequential sampling of a DPP is that we do *not* calculate the conditional kernel (3.1.4). The second difference is that the linear systems that are solved by RAS are *regularized* and Theorem 3.1.1 accounts for this regularization. Thirdly, RAS involves a *weighting* of the sampling matrix which does not appear in DPP sampling.

3.1.6 Numerical results

Exact algorithms

The performance of deterministic adaptive landmark sampling (DAS) and its randomized variant (RAS) is evaluated on the `Boston housing`, `Stock`, `Abalone` and `Bank 8FM` datasets which are described in Table 3.1.1. These

Table 3.1.1: Small scale data sets and parameters

data set	n	d	c/ϵ	σ
`Housing`	506	13	100	5
`Stock`	950	10	100	5
`Abalone`	4177	8	150	5
`Bank 8FM`	8192	8	150	10

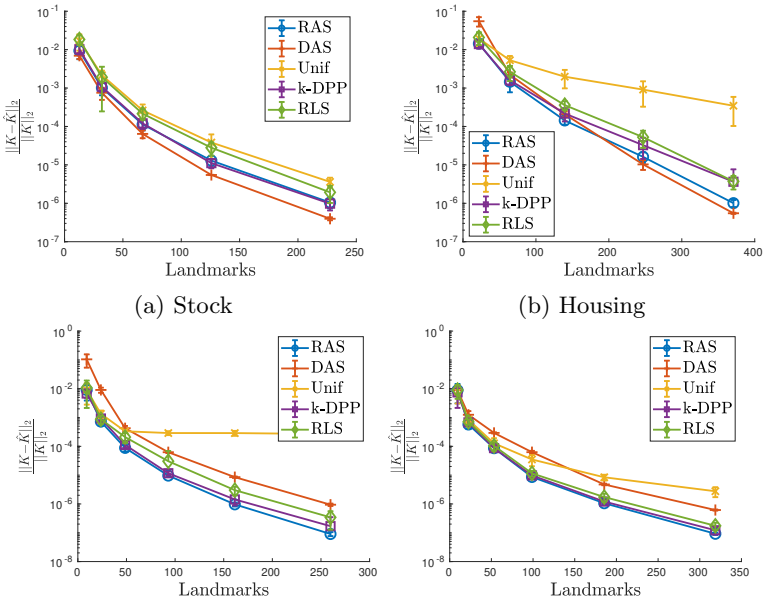
public data sets[2] have been used for benchmarking m-DPPs in [136]. The quality of the landmarks $\mathcal{C}$ is evaluated by calculating $\|K - \hat{K}\|_2/\|K\|_2$ with $\hat{K} = L_{\varepsilon,C}(K)$ with $\varepsilon = 10^{-12}$ for numerical stability. A Gaussian kernel is used with a fixed bandwidth $\sigma > 0$ after standardizing the data. We compare the following exact algorithms: Uniform sampling (Unif.), m-DPP[3], exact ridge leverage score sampling (RLS)and DAS. The experiments have the following structure:

1. First, RAS is executed for an increasing $\gamma \in \{10^0, \ldots, 10^{-4}\}$ for the `Boston housing` and `Stock` data sets, $\gamma \in \{10^0, \ldots, 10^{-5}\}$ for the `Abalone` and `Bank 8FM` data sets and $\epsilon = 10^{-10}$. With each regularization parameter γ, RAS returns a different subset of m landmarks. The same subset size m is used for the comparisons with the other methods.

2. For each subset size m, the competing methods are executed for multiple hyperparameters $\gamma \in \{10^0, 10^{-1}, \ldots, 10^{-6}\}$. The parameter γ here corresponds to the regularization parameter for the RLSs (see (3.1.2)), and the regularization parameter of DAS. The best performing γ is selected for each subset size m and sampling algorithm. In this manner, we each time compare the best possible sampling every algorithm can provide given the subset size m.

[2] `https://www.cs.toronto.edu/~delve/data/datasets.html`, `https://www.openml.org/d/223`

[3] Matlab code: `https://www.alexkulesza.com/`.



(a) Stock (b) Housing

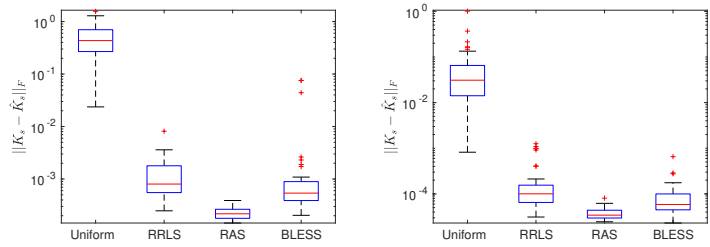
(c) Abalone (d) Bank8FM

Figure 3.1.3: Relative operator norm of the Nyström approximation error as a function of the number of landmarks. The error is plotted on a logarithmic scale, averaged over 10 trials. The error bars are standard deviations.

This procedure is repeated 10 times and the averaged results are visualized in Figure 3.1.3. Notice that the error bars are small. DAS performs well on the `Boston housing` and `Stock` data set, which show a fast decay in the spectrum of K (see Figure B.6.1 in the Appendix. This confirms the expectations from Proposition 3.1.5. If the decay of the eigenvalues is not fast enough, the randomized variant RAS has a superior performance which is similar to the performance of a m-DPP. For our implementation, RAS is faster compared with m-DPP, as it is illustrated in Figure B.6.3 in the Appendix. The main cost of RAS is the calculation of the projector kernel matrix which requires $O(n^3)$ operations. The exact sampling of a m-DPP has a similar cost. As a measure of diversity the logdet(K_{CC}) is given in Figure B.6.3 in the Appendix. The DAS method shows the largest determinant, RAS has a similar determinant as the m-DPP.

Approximate RAS

For medium-scale problems, we propose an approximate RAS method. First, the Gaussian kernel matrix K is approximated by using random Fourier features [185], that is, a generic method that is not data-adaptive. In practice, we calculate $F \in \mathbb{R}^{n \times n_F}$ with $n_F = 4000$ random Fourier features such that $FF^\top$ approximates K. Then, we obtain $\hat{P}_{n\gamma} = F\left(F^\top F + n\gamma \mathbb{I}\right)^{-1} F^\top$, where the linear system is solved by using a Cholesky decomposition. This approximate projector matrix is used in the place of $P_{n\gamma}$ in Algorithm 3.1.2. Next, we use the matrix inversion lemma in order to update the matrix inverse needed to calculate the sampling probability in Algorithm 3.1.2. The quality of the obtained landmarks is evaluated by calculating the averaged error $\|K_{\mathcal{A}\mathcal{A}} - \hat{K}_{\mathcal{A}\mathcal{A}}\|_F$ over 50 subsets $\mathcal{A} \subset [n]$ of 2000 points sampled uniformly at random. The Frobenius norm is preferred to the 2-norm to reduce the runtime. A comparison is done with the recursive leverage score sampling method[4] (RRLS) developed in [165] and BLESS which was proposed in [200]. The parameters of BLESS are chosen in order to promote accuracy rather than speed (see Table B.6.1 in the Appendix). This algorithm was run several times until the number of sampled landmarks n_{BLESS} satisfies $|\mathcal{C}| \leq n_{BLESS} \leq |\mathcal{C}| + 20$. An overview of the data sets and



(a) $|\mathcal{C}| = 2378$, $\sigma = 5$. (b) $|\mathcal{C}| = 924$, $\sigma = 10$.

Figure 3.1.4: Boxplot of the Nyström approximation error in Frobenius norm for `MiniBooNE`. The runtimes are: (a) $t_{RRLS} = 8$s, $t_{RAS} = 710$s, $t_{BLESS} = 18$s and (b) $t_{RRLS} = 3$s, $t_{RAS} = 167$s, $t_{BLESS} = 5$s.

parameters used in the experiments[5] is given in Table 3.1.2. Note that here we take a fixed σ and γ and not repeat the experiment for multiple subsets sizes $|\mathcal{C}|$.

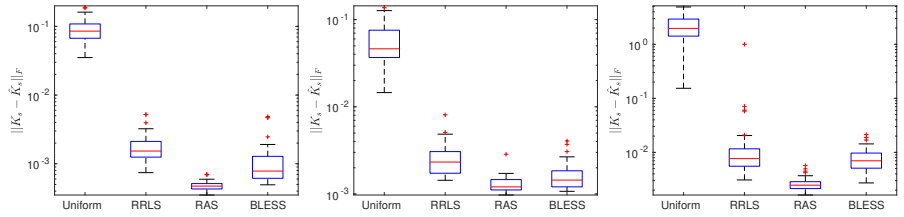
The experiments are repeated 3 times, except for `Covertype`, for which the computation is done only once. The sampling with the lowest average error over the 50 subsets is shown for each method. Hence, the error bars intuitively

[4] Matlab code: `https://www.chrismusco.com/`.
[5] `https://archive.ics.uci.edu/ml/datasets.php`

Table 3.1.2: Medium scale data sets and parameters

data set	n	d	c/ϵ	σ	γ
`Super`	21263	81	1	3	10^{-4}
`CASP`	45730	9	1	1	10^{-4}
`Adult`	48842	14	200	10	10^{-7}
`MiniBooNE`	130065	50	200	5,10	10^{-7}
`cod-RNA`	331152	8	200	4	10^{-6}
`Covertype`	581012	54	200	10	10^{-6}

indicate the variance of the stochastic estimate of the Frobenius error. In Figure 3.1.4, the sampling of landmarks with RAS for two different kernel parameters is compared to RRLS and BLESS on `MiniBooNE`. The boxplots show a reduction of the error, as well as a lower variance to the advantage of RAS. The increase in performance becomes more apparent when fewer landmarks are sampled. Additional experiments are given in Figure 3.1.5 for `Adult`, `cod-RNA` and `Covertype`. The indicative runtimes for RRLS, RAS, and BLESS are only given for completeness. The introduction of random Fourier features to approximate the projector kernel matrix significantly reduces the memory cost and computation time of the first step in the algorithm. The subsequent steps may be costly if n is large, as we have to make a full pass over the dataset.

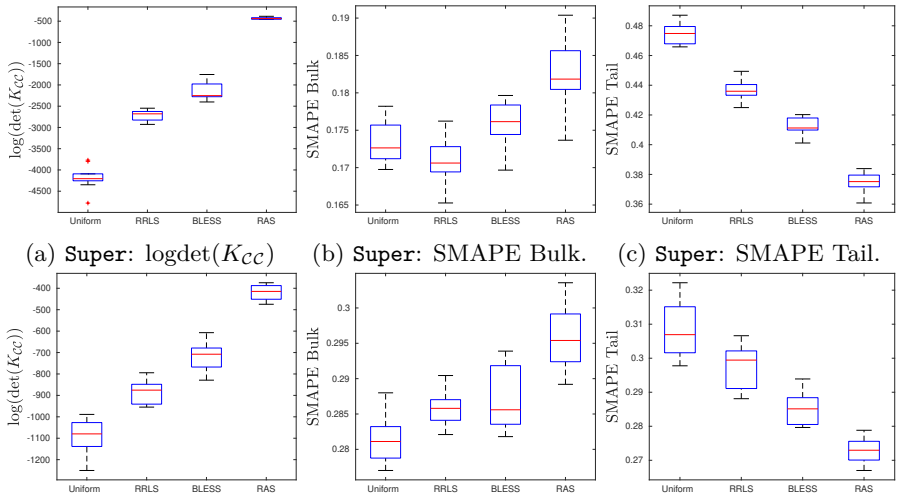


(a) `Adult`, $|\mathcal{C}| = 1816$. (b) `cod-RNA`, $|\mathcal{C}| = 1280$. (c) `Covertype`, $|\mathcal{C}| = 3016$.

Figure 3.1.5: Boxplot of the Nyström approximation error in Frobenius norm. The runtimes are: (a) $t_{RRLS} = 3$s, $t_{RAS} = 197$s, $t_{BLESS} = 4$s, (b) $t_{RRLS} = 11$s, $t_{RAS} = 555$s, $t_{BLESS} = 11$s and (c) $t_{RRLS} = 70$s, $t_{RAS} = 3131$s, $t_{BLESS} = 71$s.

Performance for Kernel Ridge Regression

To conclude, we verify the performance of the proposed method on a supervised learning task. Consider the input-output pairs $\{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i \in [n]}$. The



(a) `Super`: logdet($K_{\mathcal{C}\mathcal{C}}$) (b) `Super`: SMAPE Bulk. (c) `Super`: SMAPE Tail.

(d) `CASP`: logdet($K_{\mathcal{C}\mathcal{C}}$) (e) `CASP`: SMAPE Bulk. (f) `CASP`: SMAPE Tail.

Figure 3.1.6: Boxplot of the logdet($K_{\mathcal{C}\mathcal{C}}$), SMAPE in the bulk and tail. The runtimes are: (`CASP`) $t_{RRLS} = 0.4$s, $t_{RAS} = 55$s, $t_{BLESS} = 19$s, (`Super`) $t_{RRLS} = 0.6$s, $t_{RAS} = 68$s, $t_{BLESS} = 14$s.

Kernel Ridge Regression (KRR) using the subsample $\mathcal{C}$ is determined by solving

$$f^\star = \arg\min_{f\in\mathcal{H}_\mathcal{C}} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2 + \lambda\|f\|^2_{\mathcal{H}}, \text{ with } \lambda > 0,$$

where $\mathcal{H}_\mathcal{C} = \text{span}\{k(\boldsymbol{x}_i,\cdot)|i\in\mathcal{C}\}$. The regressor is $f^\star(\cdot) = \sum_{i\in\mathcal{C}}\alpha_i^\star k(\boldsymbol{x}_i,\cdot)$ with

$$\boldsymbol{\alpha}^\star = (K_\mathcal{C}^\top K_\mathcal{C} + n\lambda K_{\mathcal{C}\mathcal{C}})^{-1}K_\mathcal{C}^\top y. \tag{3.1.13}$$

The experiment is done on the following data sets: `Superconductivity` (`Super`) and `Physicochemical Properties of Protein Tertiary Structure` (`CASP`)[6], where we use a Gaussian Kernel after standardizing data. The symmetric mean absolute percentage error (SMAPE) between the outputs and the regressor

$$\text{SMAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - f^\star(\boldsymbol{x}_i)|}{(|y_i| + |f^\star(\boldsymbol{x}_i)|)/2},$$

is calculated for the methods: Uniform sampling, RRLS, BLESS, and approximate RAS. First, approximate RAS is run with $n_F = 4000$ random

[6] `https://archive.ics.uci.edu/ml/datasets.php`

Fourier features. Afterwards Uniform, RRLS and BLESS sample the same number of landmarks as approximate RAS. To evaluate the performance of different methods on the full space of data sets, the test set is stratified, i.e., the test set is divided into 'bulk' and 'tail' as follows. The bulk corresponds to test points where the RLSs are smaller than or equal to the 70% quantile, while the tail of data corresponds to test points where the ridge leverage score is larger than the 70% quantile. The RLSs are approximated using RRLS with 8000 data points. This way, we can evaluate if the performance of the different methods is also good for 'outlying' points. A similar approach was conducted in the previous chapter. The data set is split into 50% training data and 50% test data, so to make sure the train and test set have similar RLS distributions. Information on the data sets and hyperparameters used for the experiments is given in Table 3.1.2 in the Appendix. The regularization parameter $\lambda \in \{10^{-4}, 10^{-6}, 10^{-8}, 10^{-12}\}$ of Kernel Ridge Regression is determined by using 10-fold-cross-validation. The results in Figure 3.1.6 show that approximate RAS samples a more diverse subset. As a result, the method performs much better in the tail of the data, while yielding a slightly worse performance in the bulk. The performance gain in the tail of the data is more apparent for the `Super`, where the RLS distribution has a longer tail. The data set has more 'outliers', increasing the importance of using a diverse sampling algorithm. The RLS distribution of the data sets is visualized in Appendix B.6.

3.2 Towards deterministic diverse subset sampling

3.2.1 Introduction

Deterministic algorithms are interesting for many applications, as they provide interpretability to the user by having no chance of failure and always returning the same results. The usefulness of deterministic algorithms has already been recognized by Papailiopoulos et al. [176] and McCurdy [155], who provide deterministic algorithms based on the (ridge) leverage scores. These statistical leverage scores correspond to correlations between the singular vectors of a matrix and the canonical basis [9, 68]. The Deterministic Adaptive Sampling (DAS) algorithm of the previous section provides a deterministically obtained *diverse* subset. The method shows superior performance compared to randomized counterparts in terms of the approximation error for the Nyström approximation when the eigenvalues of the kernel matrix have a fast decay. A similar observation was made for the deterministic algorithms of Papailiopoulos et al. [176] and McCurdy [155].

This section discusses a deterministic adaption of k-DPP, where we have the following empirical observations:

1. The method is deterministic, hence there is no failure probability and the method always produces the same output.
2. Only the k eigenvectors with the largest eigenvalues are needed, which results in a speedup when $k \ll n$.
3. We observed that the method samples a more diverse subset than the original k-DPP on multiple datasets.
4. There is no need to tune a regularization parameter, which is the case for the DAS algorithm.
5. The method shows superior accuracy in terms of the max norm of the Nyström approximation on multiple datasets compared to the standard k-DPP, along with better accuracy of the kernel approximation for the operator norm when there is fast decay of the eigenvalues.

In Section 3.2.2, we introduce the method. Secondly, we make a connection with the DAS algorithm, namely the deterministic k-DPP corresponds to the DAS algorithm with an adapted projector kernel matrix. In Section 3.2.4, we evaluate the method on different datasets. Finally, a small real-life illustration is shown in Section 3.2.5.

Algorithm 3.2.1 Exact DPP sampling algorithm associated to the L-ensemble L [125]. Notice that $P = L(L + \mathbb{I})^{-1}$, so that the eigenvector/value pairs of P are exactly $\{(\boldsymbol{v}_i, \frac{\lambda_i}{\lambda_i+1})\}_{i=1}^n$.

---

input: L-ensemble $L \succeq 0$
initialization: $\mathcal{J} = \emptyset$ and $\mathcal{Y} = \emptyset$
Calculate the eigenvector/value pairs $\{(\boldsymbol{v}_i, \lambda_i)\}_{i=1}^n$ of L.
for: $i = 1, \ldots, n$ **do**
  $\mathcal{J} \leftarrow \mathcal{J} \cup \{i\}$ with prob. $\frac{\lambda_i}{\lambda_i+1}$
end for
$V \leftarrow \{\boldsymbol{v}_i\}_{i\in\mathcal{J}}$ a set of columns
while: $|V| > 0$ **do**
  Draw an index i according to the distribution $p_i = \frac{1}{|V|}\sum_{\boldsymbol{v}\in V}(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{e}_i)^2$.
  $\mathcal{Y} \leftarrow \mathcal{Y} \cup i$
  $V \leftarrow V_\perp$, orthonormal basis for the subspace of V orthogonal to $\boldsymbol{e}_i$.
end while
return $\mathcal{Y}$.

3.2.2 Deterministic adaptation of k-DPP

We discuss a deterministic adaptation of k-DPP, by selecting iteratively landmarks with the highest probability. As it is described in Algorithm 3.2.2, we can successively maximize the probability over a nested sequence of sets $\mathcal{C}_0 \subseteq \mathcal{C}_1 \subseteq \cdots \subseteq \mathcal{C}_k$ starting with $\mathcal{C}_0 = \emptyset$ by adding one landmark at each iteration. The proposed method is an adaptation of the improved k-DPP sampling algorithm given by Tremblay et al. [231]. Our proposed method start from a projective marginal kernel $P = VV^\top$, with $V = [\boldsymbol{v}_1, ..., \boldsymbol{v}_k] \in \mathbb{R}^{n\times k}$ the sampled eigenvectors of the kernel matrix. Instead of sampling the eigenvectors [125], the k eigenvectors with the largest eigenvalue are chosen. Secondly, at each iteration the point with the highest probability $p(i) = P_{ii} - P_{\mathcal{C}i}^\top P_{\mathcal{C}\mathcal{C}}^\dagger P_{\mathcal{C}i}$ is chosen, where $\mathcal{C}$ corresponds to the selected subset so far. Besides the interpretation of DPPs in relation to diversity, the aforementioned probability gives a second insight in diversity. Namely we have $p(i) = \|\boldsymbol{v}_i - \pi_{V_\mathcal{C}}\boldsymbol{v}_i\|_2^2$, where $\boldsymbol{v}_i \in \mathbb{R}^{n\times 1}$ is the i-th column of V and $\pi_{V_\mathcal{C}}$ is the projector on $V_\mathcal{C} = \text{span}\{\boldsymbol{v}_s | s \in \mathcal{C}\}$. The chosen landmark corresponds to the point that is the most distant to the space of the previously sampled points.

Algorithm 3.2.2 Deterministic adaptation of the k-DPP sampling algorithm.

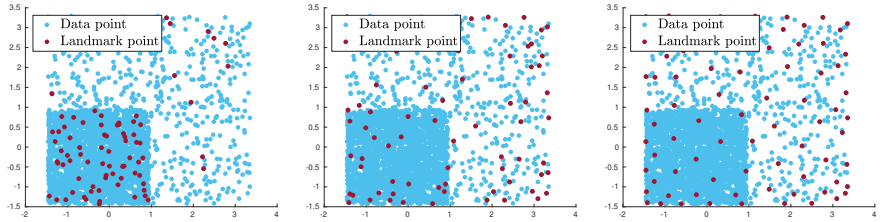
input: Kernel matrix K, sample size k.
initialization: $\mathcal{C} \leftarrow \emptyset$
Calculate the first k eigenvectors $\mathrm{V} \in \mathbb{R}^{n\times k}$ from K.
$P = VV^{\mathrm{T}}$
Define $p_0 \in \mathbb{R}^N : \forall i, \quad p_0(i) = \left\|\mathrm{V}^\top \boldsymbol{e}_i\right\|^2$
$p \leftarrow p_0$
for: $i = 1, \ldots, k$ **do**
  Select c_i with highest probability $p(i)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_i\}$
  Update $\boldsymbol{p} : \forall j \quad p(j) = p_0(j) - P_{\mathcal{C}j}^\top P_{\mathcal{C}\mathcal{C}}^\dagger P_{\mathcal{C}j}$
end for
return $\mathcal{C}$.

3.2.3 Connections with DAS

Algorithmically, the proposed method corresponds to the DAS algorithm with a different projector kernel matrix. More precisely, DAS uses a smoothed projector kernel matrix $P_{n\gamma}(K) = K(K + n\gamma I)^{-1}$ with $K \succ 0$. The ridge leverage scores $l_i(\gamma) = \sum_{j=1}^n \frac{\lambda_j}{\lambda_j + n\gamma} V_{ij}^2$ can be found on the diagonal $P_{n\gamma}(K)$. Let $V \in \mathbb{R}^{n\times n}$ be the matrix of eigenvectors of the kernel matrix K. On the contrary, in

this section, the proposed method has a *sharp projector kernel matrix* with the rank-k leverage scores $l_i = \sum_{j=1}^{k} V_{ij}^2$ on the diagonal. This has the added benefit that there is no regularization parameter to tune.

Algorithm 3.2.2 is a greedy reduced basis method as defined in [61]. Other greedy maximum volume approaches are found in [40, 46, 85]. These greedy methods are used for finding an estimate of the most likely configuration (MAP), which is known to be NP-hard [85]. In practice, we see that the method performs quite well and gives a consistently larger $\det(K_{\mathcal{CC}})$, which is considered a measure for diversity, compared to the randomized counterpart (see Section 3.2.4). This number is calculated as follows $\log(\det(K_{\mathcal{CC}})) = \sum_{i=1}^{k} \log(\lambda_i)$, with $\{\lambda_i\}_{i=1}^{k}$ the singular values of $K_{\mathcal{CC}}$. A small illustration is given in Figure 3.2.1, where the deterministic algorithm gives a more diverse subset.



(a) Uniform (b) k-DPP (c) Deterministic k-DPP

Figure 3.2.1: Illustration of sampling methods on an artificial dataset. Uniform sampling does not promote diversity and selects almost all points in the bulk of the data. Sampling a k-DPP overcomes this limitation, however, landmarks can be close to each other. The latter is solved by using the deterministic adaptation of k-DPP, which gives a more diverse subset.

3.2.4 Numerical results

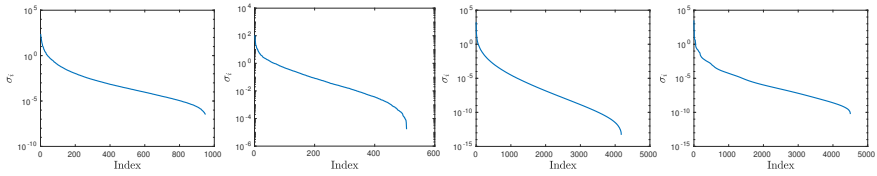
We evaluate the performance of the deterministic variant of the k-DPP with a Gaussian kernel on the `Boston housing`, `Stock`, `Abalone` and `Bank 8FM` datasets, which have 506, 950, 4177 and 8192 datapoints respectively. Those public datasets[7] have been used for benchmarking k-DPPs in [136]. The implementation of the algorithms is done with MatlabR2018b.

Throughout the experiments, we use a fixed bandwidth $\sigma = 2$ for `Boston housing`, `Stock` and `Abalone` dataset and $\sigma = 5$ for the `Bank 8FM` dataset after standardizing the data. The following algorithms are used to sample k landmarks:

[7] `https://www.cs.toronto.edu/~delve/data/datasets.html`, `https://www.openml.org/d/223`

Uniform sampling, k-DPP[8] [124], DAS and the proposed method. DAS is executed for multiple regularization parameters $\gamma \in \{10^0, 10^{-1}, \ldots, 10^{-6}\}$ where the sample with the best performing γ is selected to approximate the kernel matrix. The total experiment is repeated 10 times. The quality of the landmarks $\mathcal{C}$ is evaluated by the relative operator or max norm $\|K - \hat{K}\|_{\{\infty,2\}} / \|K\|_{\{\infty,2\}}$ with $\hat{K} = K_{\mathcal{C}}(K_{\mathcal{C}\mathcal{C}} + \epsilon\mathbb{I})^{-1}K_{\mathcal{C}}^{\top}$ with $\varepsilon = 10^{-12}$ for numerical stability. The max norm and the operator norm of a matrix A are given respectively by $\|A\|_{\infty} = \max_{i,j} |A_{ij}|$ and $\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2$. The diversity is measured by $\log(\det(K_{\mathcal{C}\mathcal{C}}))$, where a larger log determinant means more diversity. The results are visible in Figure 3.2.6. The computer used for these simulations has 8 processors $3.40GHz$ and 15.5 GB of RAM.

As previously mentioned, the greedy method returns a more diverse subset. Figure 3.2.6 shows the $\log(\det(K_{\mathcal{C}\mathcal{C}}))$ where the proposed method shows similar performance as DAS, while improving on both the randomized methods. The same is visible for the relative max norm of the Nyström approximation error. DAS and the deterministic variant of the k-DPP perform well on the `Boston housing` and `Stock` dataset, which show a fast decay in the spectrum of K (see Figure 3.2.2). If the decay of the eigenvalues is not fast enough, the randomized k-DPP, shows better performance. The same observation was made for the deterministic (ridge) leverage score sampling algorithms [155, 176] as well as DAS.



(a) `Stock` (b) `Housing` (c) `Abalone` (d) `Bank8FM`

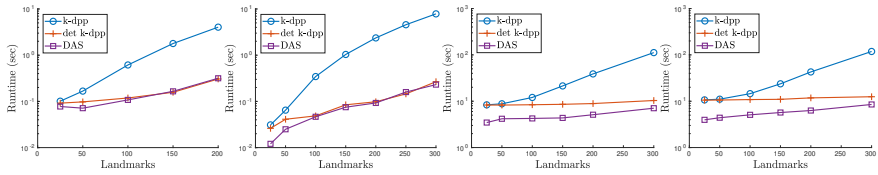
Figure 3.2.2: Singular value spectrum of the datasets on a logarithmic scale. For a given index, the value of the eigenvalues for the `Stock` and `Housing` dataset are smaller than `Abalone` and `Bank8FM`.

3.2.5 Illustration

We demonstrate the use of the proposed method on an image summarization task. The first experiment is done on the `Stanford Dogs` dataset[9], which contains images of 120 breeds of dogs from around the world. This dataset

[8] We used the Matlab code available at `https://www.alexkulesza.com/`.

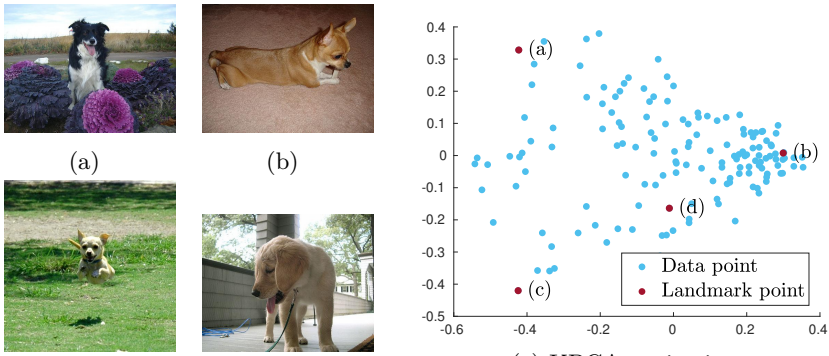
[9] `http://vision.stanford.edu/aditya86/ImageNetDogs/main.html`



(a) `Stock` (b) `Housing` (c) `Abalone` (d) `Bank8FM`

Figure 3.2.3: Timings for the computations of Figure 3.2.6 as a function of the number of landmarks. The timings are plotted on a logarithmic scale, averaged over 10 trials.

has been built using images and annotation from ImageNet for the task of fine-grained image categorization. The training features are SIFT descriptors [148] (given by the dataset), which are used to make a histogram intersection kernel. We take a subset of 50 images each of the classes *border collie*, *chihuahua*, and *golden retriever*. Figure 3.2.4 displays the results of the method for $k = 4$. One can observe that the images are very dissimilar and dogs out of each breed are represented. This is confirmed by the projection of the landmarks on the 2 first principal components of the kernel principal component analysis (KPCA) [207], where the landmark points lie in the outer regions of the space.



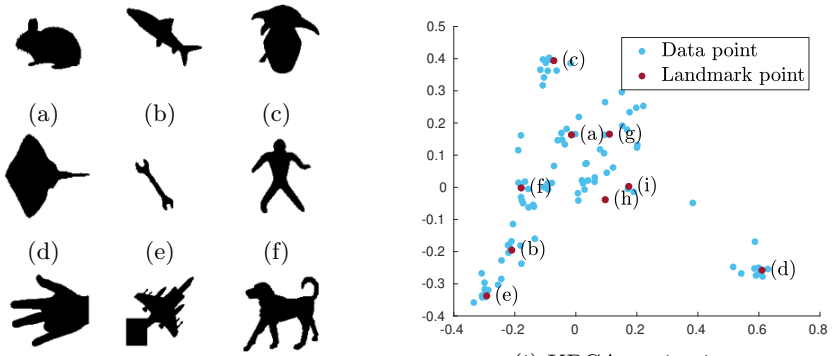
(a) (b) (c) (d) (e) KPCA projection

Figure 3.2.4: Illustration of the proposed method with $k = 4$ on the `Stanford Dogs` dataset. The selected landmark points are visualized on the left, the projection on the 2 first principal components of the KPCA on the right.

We repeat the above procedure on the `Kimia99` dataset[10]. The dataset has

[10] `https://vision.lems.brown.edu/content/available-software-and-databases#Datasets-Shape`

9 classes consisting of 11 images each. It contains shapes silhouettes for the classes: rabbits, quadrupeds, men, airplanes, fish, hands, rays, tools, and a miscellaneous class. First, we resize the images to size 100×100. Afterwards, we apply a Gaussian kernel with bandwidth $\sigma = 100$ after standardizing the data. The results of the proposed method with $k = 9$ are visible in Figure 3.2.5. The method samples landmarks out every class, making it a desirable image summarization. This is supported by the projection of the landmarks on the 2 first principal components of the KPCA, where a landmark point is chosen out of every small cluster.



(j) KPCA projection

Figure 3.2.5: Illustration of the proposed method with $k = 9$ on the `Kimia99` dataset. The selected landmark points are visualized on the left, the projection on the 2 first principal components of the KPCA on the right.

3.3 Conclusion

In this chapter, we discussed 3 new methods that promote diversity sampling. Motivated by the connection between leverage scores, DPPs, and kernelized Christoffel functions, we proposed two sampling algorithms: DAS and RAS, with a theoretical analysis. RAS allows to sample diverse and important landmarks with a good overall performance. An additional approximation is proposed so that RAS can be applied to large data sets in the case of a Gaussian kernel. Experiments for kernel ridge regression show that the proposed method performs much better in the tail of the data, while maintaining comparable performance for the bulk data. Finally, we discussed a greedy deterministic adaptation of k-DPP. Algorithmically, the method corresponds to the DAS algorithm with

a different projector kernel matrix. The proposed method is evaluated by comparing the accuracy of the Nyström approximation on multiple datasets. Experiments show the proposed method is able to give a more diverse subset, along with better performance for the relative max norm. When there is a fast decay of the eigenvalues, the deterministic method is more accurate than the randomized counterparts.

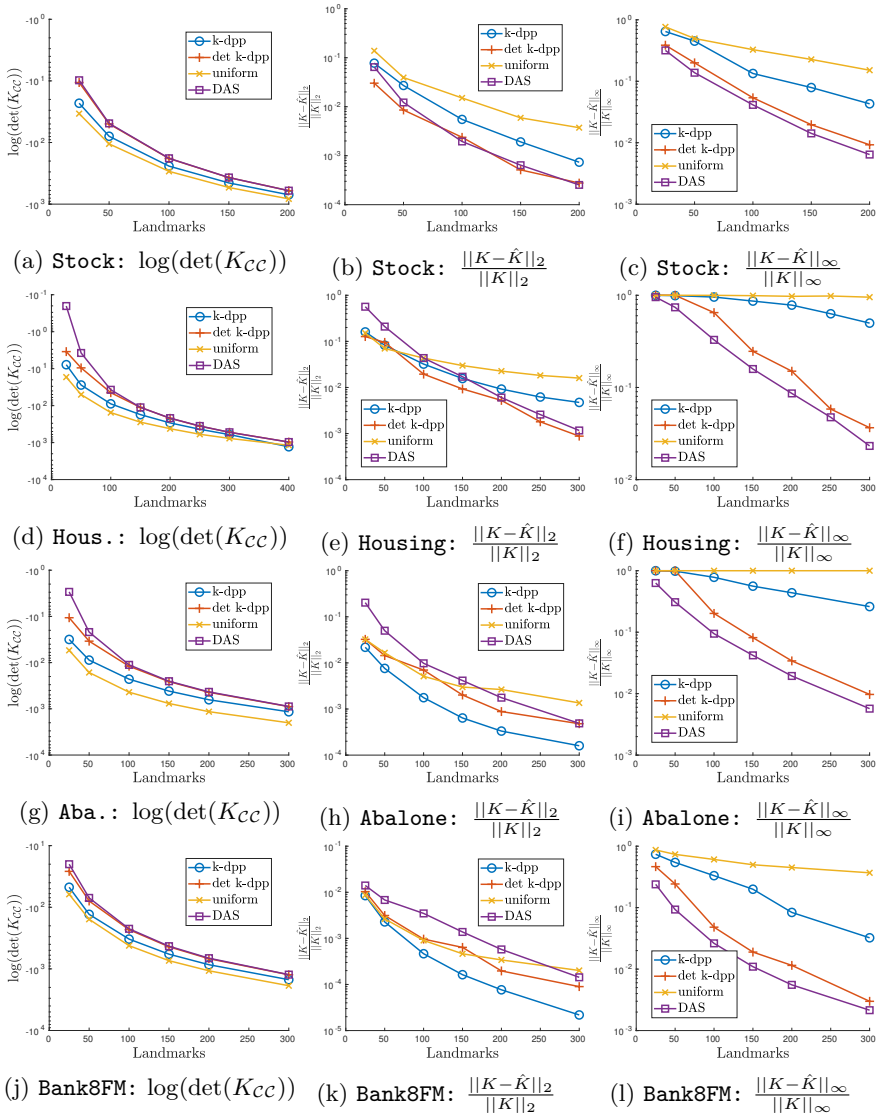


Figure 3.2.6: $\log(\det(K_{\mathcal{CC}}))$, relative operator norm, and relative max norm in function of the number of landmarks. The error is plotted on a logarithmic scale, averaged over 10 trials.

Chapter 4

Diversity in generative adversarial networks

This chapter comprises previously published articles including:

- Hannes De Meulenmeester, Joachim Schreurs[1], Michaël Fanuel, Bart De Moor, and Johan AK Suykens. The Bures Metric for Generative Adversarial Networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2021.
- Joachim Schreurs[2], Hannes De Meulenmeester, Michaël Fanuel, Bart De Moor, and Johan AK Suykens. Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks. In *International Conference on Machine Learning, Optimization, and Data Science (LOD)*, 2021.

[1]The candidate helped write the article, performed part of the experiments, and defined the methodology in collaboration with the coauthors.

[2]The candidate performed the research and wrote the article in collaboration with the coauthors.

4.1 The Bures metric for generative adversarial networks

4.1.1 Introduction

Generative Adversarial Networks (GANs) [88] are particularly successful approaches to produce high-quality samples, especially in the case of natural images, though their training is notoriously difficult. The vanilla GAN consists of two networks: a generator and a discriminator. The generator maps random noise, usually drawn from a multivariate normal, to fake data in input space. The discriminator estimates the likelihood ratio of the generator network to the data distribution. A common problem in GANs is that the generator only generates samples from a few of the many modes of the distribution. This phenomenon is called 'mode collapse'.

In this section we propose BuresGAN: a generative adversarial network that has the objective function of a vanilla GAN complemented by an additional term, given by the squared Bures distance between the covariance matrix of real and fake batches in a latent space. This loss function promotes matching of fake and real data in a feature space $\mathbb{R}^f$, so that mode collapse is reduced. Conveniently, the Bures distance also admits both a feature space and kernel-based expression. Contrary to other related approaches such as in [38] or [217], the architecture of the GAN is unchanged, only the objective is modified. We empirically show that the proposed method is robust when it comes to the choice of architecture and does not require an additional fine architecture search. Finally, an extra asset of BuresGAN is that it yields competitive or improved IS and FID scores compared with the state of the art on CIFAR-10 and STL-10 using a ResNet architecture.

Related works. The Bures distance is closely related to the Fréchet distance [67] which is a 2-Wasserstein distance between multivariate normal distributions. Namely, the Fréchet distance between multivariate normals of equal means is the Bures distance between their covariance matrices. The Bures distance is also equivalent to the exact expression for the 2-Wasserstein distance between two elliptically contoured distributions with the same mean as shown in [81] and [179]. Noticeably, it is also related to the Fréchet Inception Distance score (FID), which is a popular manner to assess the quality of generative models. This score uses the Fréchet distance between real and generated samples in the feature space of a pre-trained inception network as it is explained in [204] and [98].

There exist numerous works aiming to improve the training efficiency of generative networks. For mode collapse evaluation, we compare BuresGAN to the most closely related works. GDPP-GAN [74] and VEEGAN [217] also try to enforce diversity in 'latent' space. GDPP-GAN matches the eigenvectors and eigenvalues of the real and fake diversity kernel. In VEEGAN, an additional reconstructor network is introduced to map the true data distribution to Gaussian random noise. In a similar way, architectures with two discriminators are analyzed by [170], while MADGAN [83] uses multiple discriminators and generators. A different approach is taken by Unrolled-GAN [159] which updates the generator with respect to the unrolled optimization of the discriminator. This allows training to be adjusted between using the optimal discriminator in the generator's objective, which is ideal but infeasible in practice. Wasserstein GANs [12, 91] leverage the 1-Wasserstein distance to match the real and generated data distributions. In MDGAN [38], a regularization is added to the objective function, so that the generator can take advantage of another similarity metric with a more predictable behavior. This idea is combined with a penalization of the missing modes. Some other recent approaches to reducing mode collapse are variations of WGAN [247]. Entropic regularization has been also proposed in PresGAN [62], while metric embeddings were used in the paper introducing BourGAN [250]. A simple packing procedure that significantly reduces mode collapse was proposed in PacGAN [141] that we also consider hereafter in our comparisons.

4.1.2 Method

A GAN consists of a discriminator $D : \mathbb{R}^d \to \mathbb{R}$ and a generator $G : \mathbb{R}^\ell \to \mathbb{R}^d$ which are typically defined by neural networks and parametrized by real vectors. The value $D(\boldsymbol{x})$ gives the probability that $\boldsymbol{x}$ comes from the empirical distribution, while the generator G maps a point $\boldsymbol{z}$ in the latent space $\mathbb{R}^\ell$ to a point in the input space $\mathbb{R}^d$. The training of a GAN consists in solving

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))], \tag{4.1.1}$$

by alternating two phases of training. In (4.1.1), the expectation in the first term is over the empirical data distribution p_d, while the expectation in the second term is over the generated data distribution p_g, implicitly given by the mapping by G of the latent prior distribution $\mathcal{N}(0, \mathbb{I}_\ell)$. It is common to define and minimize the discriminator loss by

$$V_D = -\mathbb{E}_{\boldsymbol{x}\sim p_d}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x}))]. \tag{4.1.2}$$

In practice, it is proposed in [88] to minimize generator loss

$$V_G = -\mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\mathbb{I}_\ell)}[\log D(G(\boldsymbol{z}))], \tag{4.1.3}$$

rather than the second term of (4.1.1), for an improved training efficiency.

Matching real and fake data covariance. To prevent mode collapse, we encourage the generator to sample fake data of similar diversity to real data. This is achieved by matching the sample covariance matrices of real and fake data respectively. Covariance matching and similar ideas were explored for GANs in [164] and [74]. In order to compare covariance matrices, we propose to use the squared Bures distance between positive semi-definite $\ell \times \ell$ matrices [19], i.e.,

$$\begin{aligned}\mathcal{B}(A,B)^2 &= \min_{U \in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 \\ &= \mathrm{Tr}(A + B - 2(A^{\frac{1}{2}}BA^{\frac{1}{2}})^{\frac{1}{2}}).\end{aligned}$$

Being a Riemannian metric on the manifold of positive semi-definite matrices [153], the Bures metric is adequate to compare covariance matrices. The covariances are defined in a feature space associated to the discriminator. Namely, let $D(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \phi(\boldsymbol{x}))$, where $\boldsymbol{w}$ is the weight vector of the last dense layer and σ is the sigmoid function. The last layer of the discriminator, denoted by $\phi(\boldsymbol{x}) \in \mathbb{R}^{d_\phi}$, naturally defines a feature map. We use the normalization $\bar{\phi}(\boldsymbol{x}) = \phi(\boldsymbol{x})/\|\phi(\boldsymbol{x})\|_2$, after the centering of $\phi(\boldsymbol{x})$. Then, we define a covariance matrix as follows: $C(p) = \mathbb{E}_{\boldsymbol{x}\sim p}[\bar{\phi}(\boldsymbol{x})\bar{\phi}(\boldsymbol{x})^\top]$. For simplicity, we denote the real data and generated data covariance matrices by $C_d = C(p_d)$ and $C_g = C(p_g)$, respectively. Our proposal is to replace the generator loss by $V_G + \lambda\mathcal{B}(C_d, C_g)^2$. The value $\lambda = 1$ was found to yield good results in the studied data sets. Two specific training algorithms are proposed. Algorithm 4.1.1 deals with the squared Bures distance as an additive term to the generator loss, while an alternating training is discussed in Appendix C and does not introduce an extra parameter. The training described in Algorithm 4.1.1 is analogous to

Algorithm 4.1.1 BuresGAN

Sample a real and fake batch
Update G by minimizing $V_G + \lambda\mathcal{B}(\hat{C}_r, \hat{C}_g)^2$
Update D by maximizing $-V_D$;

the training of GDPP GAN, although the additional generator loss is rather different. The computational advantage of the Bures distance is that it admits two expressions that can be evaluated numerically in a stable way. Namely, there is no need to calculate a gradient update through an eigendecomposition.

Feature space expression. In the training procedure, real $\boldsymbol{x}_i^{(d)}$ and fake data $\boldsymbol{x}_i^{(g)}$ with $1 \leq i \leq b$ are sampled respectively from the empirical distribution and the mapping of the normal distribution $\mathcal{N}(0, \mathbb{I}_\ell)$ by the generator. Consider the case where the batch size b is larger than the feature space dimension. Let the embedding of the batches in feature space be $\Phi_\alpha = [\phi(\boldsymbol{x}_1^{(\alpha)}), \ldots, \phi(\boldsymbol{x}_b^{(\alpha)})]^\top \in \mathbb{R}^{b \times d_\phi}$ with $\alpha = d, g$. The covariance matrix of one batch in feature space[1] is $\hat{C} = \bar{\Phi}^\top \bar{\Phi}$, where $\bar{\Phi}$ is the ℓ_2-normalized centered feature map of the batch. Numerical instabilities can be avoided by adding a small number, e.g. 1e−14, to the diagonal elements of the covariance matrices, so that, in practice, we only deal with strictly positive definite matrices. From the computational perspective, an interesting alternative expression for the Bures distance is given by

$$\mathcal{B}\left(C_d, C_g\right)^2 = \mathrm{Tr}\big(C_d + C_g - 2(C_g C_d)^{\frac{1}{2}}\big), \tag{4.1.4}$$

whose computation requires only one matrix square root. This identity can be obtained from Lemma 4.1.1. Note that an analogous result is proved in [171].

Lemma 4.1.1. *Let A and B be symmetric positive semidefinite matrices of the same size and let $B = Y^\top Y$. Then, we have: (i) AB is diagonalizable with nonnegative eigenvalues, and (ii)* $\mathrm{Tr}((AB)^{\frac{1}{2}}) = \mathrm{Tr}((YAY^\top)^{\frac{1}{2}})$.

Proof. (i) is a consequence of Corollary 2.3 in [105]. (ii) We now follow [171]. Thanks to (i), we have $AB = PDP^{-1}$ where D is a nonnegative diagonal and the columns of P contain the right eigenvectors of AB. Therefore, $\mathrm{Tr}((AB)^{1/2}) = \mathrm{Tr}(D^{1/2})$. Then, $YAY^\top$ is clearly diagonalizable. Let us show that it shares its nonzero eigenvalues with AB. a) We have $ABP = PD$, so that, by multiplying of the left by Y, it holds that $(YAY^\top)YP = YPD$. b) Similarly, suppose that we have the eigenvalue decomposition $YAY^\top Q = Q\Lambda$. Then, we have $BAY^\top Q = Y^\top Q\Lambda$ with $B = Y^\top Y$. This means that the non-zero eigenvalues of $YAY^\top$ are also eigenvalues of BA. Since A and B are symmetric, this completes the proof. □

Kernel-based expression. Alternatively, if the feature space dimension f is larger than the batch size b, it is more efficient to compute $\mathcal{B}(\hat{C}_d, \hat{C}_g)$ thanks to $b \times b$ kernel matrices: $K_d = \bar{\Phi}_d \bar{\Phi}_d^\top$, $K_g = \bar{\Phi}_g \bar{\Phi}_g^\top$ and $K_{dg} = \bar{\Phi}_d \bar{\Phi}_g^\top$. Then, we have the kernel-based expression

$$\mathcal{B}(\hat{C}_d, \hat{C}_g)^2 = \mathrm{Tr}\left(K_d + K_g - 2\left(K_{dg} K_{dg}^\top\right)^{\frac{1}{2}}\right), \tag{4.1.5}$$

which allows to calculate the Bures distance between covariance matrices by computing a matrix square root of a $b \times b$ matrix. This is a consequence of Lemma 4.1.2.

[1] For simplicity, we omit the normalization by $\frac{1}{b-1}$ in front of the covariance matrix.

Lemma 4.1.2. *The matrices $X^\top XY^\top Y$ and $YX^\top XY^\top$ are diagonalizable with nonnegative eigenvalues and share the same non-zero eigenvalues.*

Proof. The result follows from Lemma 1 and its proof, where $A = X^\top X$ and $B = Y^\top Y$. □

Connection with Wasserstein GAN and integral probability metrics. The Bures distance is proportional to the 2-Wasserstein distance $\mathcal{W}_2$ between two elliptically contoured distributions with the same mean [81]. For instance, in the case of multivariate normal distributions, we have

$$\mathcal{B}(A,B)^2 = \min_{\pi} \mathbb{E}_{(X,Y)\sim\pi}\|X-Y\|_2^2 \text{ s.t. } \begin{cases} X \sim \mathcal{N}(0,A) \\ Y \sim \mathcal{N}(0,B), \end{cases}$$

where the minimization is over the joint distributions π. More precisely, in this section, we make the approximation that the implicit distribution of the real and generated data in the feature space $\mathbb{R}^{d_\phi}$ (associated to $\phi(x)$) are elliptically contoured with the same mean. Under different assumptions, the Generative Moment Matching Networks [188, 137] work in the same spirit, but use a different approach to match covariance matrices. On the contrary, WGAN uses the Kantorovich dual formula for the 1-Wasserstein distance: $\mathcal{W}_1(\alpha,\beta) = \sup_{f\in\text{Lip}} \int f \,\mathrm{d}(\alpha-\beta)$, where α,β are signed measures. Generalizations of such integral formulae are called integral probability metrics (see for instance [22]). Here, f is the discriminator, so that the maximization over Lipschitz functions f plays the role of the maximization over discriminator parameters in the min-max game of (4.1.1). Then, in the training procedure, this maximization alternates with a minimization over the generator parameters.

We can now discuss the connection with Wasserstein GAN. Coming back to the definition of BuresGAN, we can now explain that the 2-Wasserstein distance provides an upper bound on an integral probability metric. Then, if we assume that the densities are elliptically contoured distributions in feature space, the use of the Bures distance to calculate $\mathcal{W}_2$ allows to spare the maximization over the discriminator parameters – and this motivates why the optimization of $\mathcal{B}$ only influences updates of the generator in Algorithm 4.1.1. Going more into detail, the 2-Wasserstein distance between two probability densities (w.r.t. the same measure) is equivalent to a Sobolev dual norm, which can be interpreted as an integral probability metric. Indeed, let the Sobolev semi-norm $\|f\|_{H^1} = (\int \|\nabla f(x)\|^2 \,\mathrm{d}x)^{1/2}$. Then, its dual norm over signed measures is defined as $\|\nu\|_{H^{-1}} = \sup_{\|f\|_{H^1}\leq 1} \int f \,\mathrm{d}\nu$. It is then shown in [180] and [179] that there exist two positive constants c_1 and c_2 such that

$$c_1\|\alpha-\beta\|_{H^{-1}} \leq \mathcal{W}_2(\alpha,\beta) \leq c_2\|\alpha-\beta\|_{H^{-1}}.$$

Hence, the 2-Wasserstein distance gives an upper bound on an integral probability metric.

Algorithmic details. The matrix square root in (4.1.4) and (4.1.5) is obtained thanks to the Newton-Schultz algorithm which is inversion free and can be efficiently calculated on GPUs since it involves only matrix products. In practice, we found 15 iterations of this algorithm to be sufficient for the small-scale data sets, while 20 iterations were used for the ResNet examples. A small regularization term 1e−14 is added to the matrix diagonal for stability. The latent prior distribution is $\mathcal{N}(0, \mathbb{I}_\ell)$ with $\ell = 100$ and the parameter in Algorithm 4.1.1 is always set to $\lambda = 1$. In the tables hereafter, we indicate the largest scores in bold, although we invite the reader to also consider the standard deviation.

4.1.3 Empirical evaluation of mode collapse

BuresGAN's performances on synthetic data, artificial and real images are compared with the standard DCGAN [204], WGAN-GP, MDGAN, Unrolled GAN, VEEGAN, GDPP, and PacGAN. We want to emphasize that the purpose of this experiment is not to challenge these baselines but to report the improvement obtained by adding the Bures metric to the objective function. It would be straightforward to add the Bures loss to other GAN variants, as well as most GAN architectures, and we would expect an improvement in mode coverage and generation quality. In the experiments, we notice that adding the Bures loss to the vanilla GAN already significantly improves the results.

A low dimensional feature space ($d_\phi = 128$) is used for the synthetic data so that the feature space formula in (4.1.4) is used, while the dual formula in (4.1.5) is used for the image data sets (Stacked MNIST, CIFAR-10, CIFAR-100, and STL-10) for which the feature space is larger than the batch size. The architectures used for the image data sets are based on DCGAN [184], while results using ResNets are given in Section 4.1.4. All images are scaled in between -1 and 1 before running the algorithms. Additional information on the architectures and data sets is given in SM. The hyperparameters of other methods are typically chosen as suggested in the authors' reference implementation. The number of unrolling steps in Unrolled GAN is chosen to be 5. For MDGAN, both versions are implemented but only MDGAN-v2 gives interesting results. The first version, which corresponds to the mode regularizer, has hyperparameters $\lambda_1 = 0.2$ and $\lambda_2 = 0.4$, for the second version, which corresponds to manifold diffusion training for regularized GANs, has $\lambda = 10^{-2}$. WGAN-GP uses $\lambda = 10.0$ and $n_{\text{critic}} = 5$. All models are trained using Adam [121] with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning

rate 10^{-3} for both the generator and discriminator. Unless stated otherwise, the batch size is 256. Examples of random generations of all the GANs are given in SM. Notice that in this section we report the results achieved only at the end of the training.

Artificial data

Synthetic. RING is a mixture of eight two-dimensional isotropic Gaussians in the plane with means $2.5\times(\cos((2\pi/8)i), \sin((2\pi/8)i))$ and std 0.01 for $1 \leq i \leq 8$. GRID is a mixture of 25 two-dimensional isotropic normals in the plane with means separated by 2 and with a standard deviation 0.05. All models have the same architecture, with $\ell = 256$ following [74], and are trained for 25k iterations. The evaluation is done by sampling 3k points from the generator network. A sample is counted as high quality if it is within 3 standard deviations of the nearest mode. The experiments are repeated 10 times for all models and their performance is compared in Table 4.1.1.

BuresGAN consistently captures all the modes and produces the highest quality samples. In Figure 4.1.1 and Figure 4.1.2, we provide snapshots of generated samples for the methods studied in the case of RING and GRID. In both cases, BuresGAN seems to generate samples in such a way that the symmetry of the training data is preserved during training. This is also true for WGAN and MDGAN, while some other GANs seem to recover the symmetry of the problem at the end of the training phase. Although BuresGAN training times are larger than most other methods for this low dimensional example, we show in Appendix C that BuresGAN scales better with the input data dimension and architecture complexity.

Stacked MNIST. The Stacked MNIST data set is specifically constructed to contain 1000 known modes. This is done by stacking three digits, sampled uniformly at random from the original MNIST data set, each in a different channel. BuresGAN is compared to the other methods and is trained for 25k iterations. For the performance evaluation, we follow [159] and use the following metrics: the number of captured modes measures mode collapse and the KL divergence, which also measures sample quality. The mode of each generated image is identified by using a standard MNIST classifier which is trained up to 98.43% accuracy on the validation set (see Appendix C.4), and classifies each channel of the fake sample. The same classifier is used to count the number of captured modes. The metrics are calculated based on 10k generated images for all the models. Generated samples from BuresGAN are given in Figure 4.1.3.

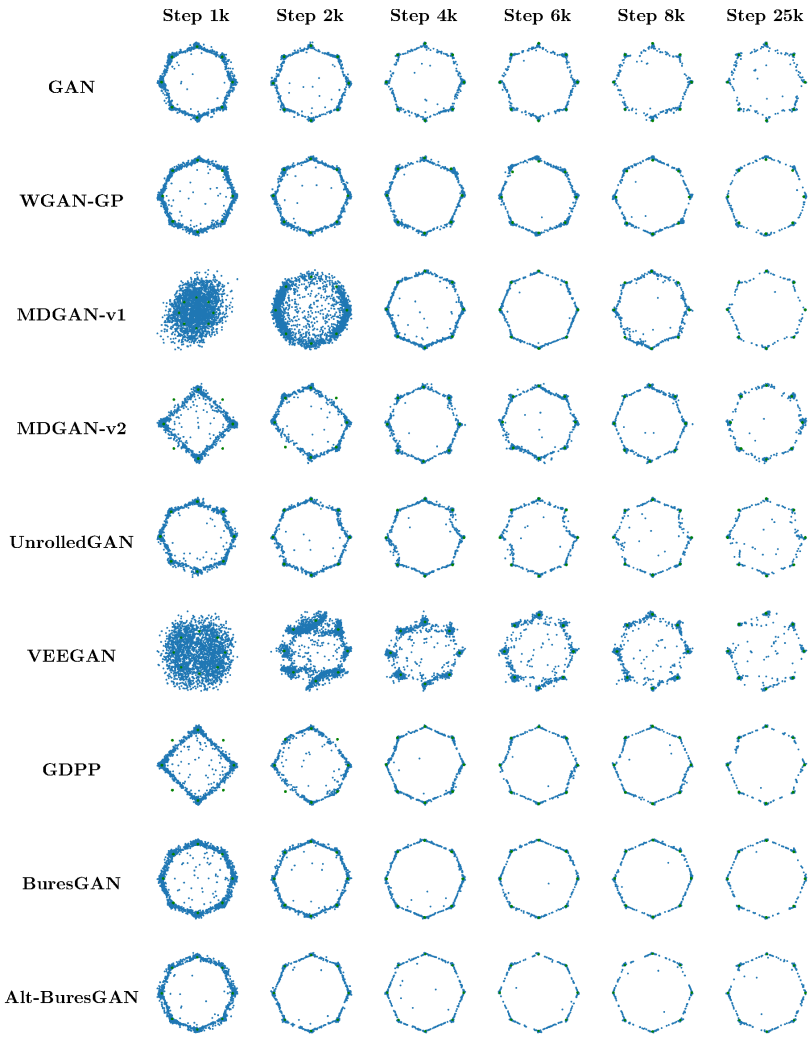


Figure 4.1.1: The progress of different GANs on the synthetic ring example. Each column shows 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

As it was observed by previous papers such as in [141] and [250], even the vanilla GAN can achieve an excellent mode coverage for certain architectures. To study mode collapse on this data set, we performed extensive experiments for multiple discriminator layers and multiple batch sizes. In the main section, the results

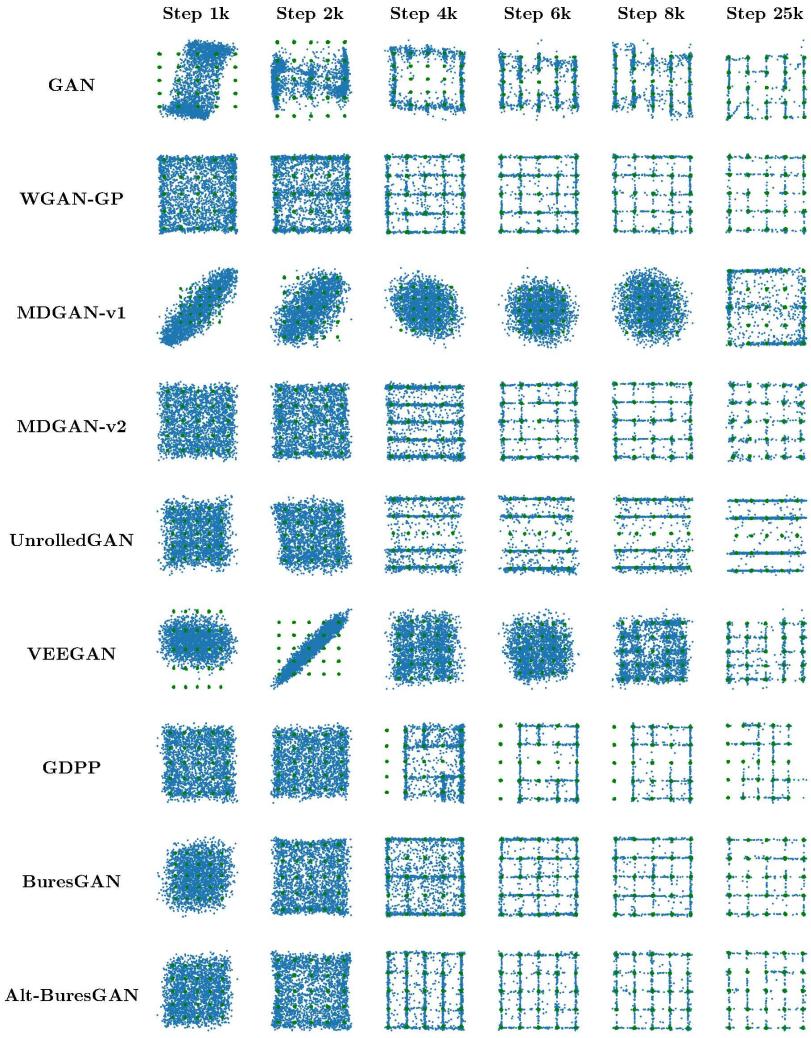


Figure 4.1.2: The progress of different GANs on the synthetic grid example. Each column shows 3000 samples from the training generator in blue with 3000 samples from the true distribution in green.

for a 3 layer discriminator are reported in Table 4.1.2. Additional experiments can be found in SM. An analogous experiment as in VEEGAN [217] with a challenging architecture including 4 convolutional layers for both the generator and discriminator was also included for completeness; see Table 4.1.3. Since

Table 4.1.1: Experiments on the synthetic data sets. Average (std) over 10 runs. All the models are trained for 25k iterations.

	Grid (25 modes)		Ring (8 modes)	
	Nb modes	% in 3σ	Nb modes	% in 3σ
GAN	22.9(4)	76(13)	7.4(2)	76(25)
WGAN-GP	24.9(0.3)	77(10)	7.1(1)	9(5)
MDGAN-v2	**25**(0)	68(11)	5(3)	20(15)
Unrolled	19.7(1)	78(19)	**8**(0)	77(18)
VEEGAN	**25**(0)	67(3)	**8**(0)	29(5)
GDPP	20.5(5)	79(23)	7.5(0.8)	73(25)
PacGAN2	23.6(4)	65(28)	**8**(0)	81(15)
BuresGAN (ours)	**25**(0)	**82**(1)	**8**(0)	**82**(4)

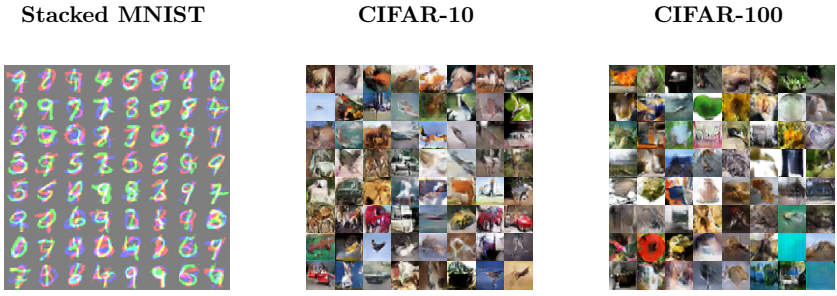


Figure 4.1.3: Generated samples from a trained BuresGAN, with a DCGAN architecture.

different authors, such as in the PacGAN's paper, use slightly different setups, we also report in Appendix C the specifications of the different settings. We want to emphasize the interest of this simulation is to compare GANs in the same consistent setting, while the results may vary from those reported in e.g. in [217] since some details might differ.

Interestingly, for most models, an improvement is observed in the quality of the images – KL divergence – and in terms of mode collapse – number of modes attained – as the size of the batch increases. For the same batch size, architecture and iterations, the image quality is improved by BuresGAN, which consistently performs well over all settings. The other methods show a higher

Table 4.1.2: KL-divergence between the generated distribution and true distribution for an architecture with 3 conv. layers for the Stacked MNIST dataset. The number of counted modes assesses mode collapse. The results are obtained after 25k iterations and we report the average(std) over 10 runs.

		Nb modes(↑)			KL div.(↓)		
	Batch size	64	128	256	64	128	256
3 conv. layers	GAN	993.3(3.1)	995.4(1.7)	**998.3**(1.2)	0.28(0.02)	**0.24**(0.02)	0.21(0.02)
	WGAN-GP	980.2(57)	838.3(219)	785.1(389)	**0.26**(0.34)	1.05(1)	1.6(2.4)
	MDGAN-v1	233.8(250)	204.0(202)	215.5(213)	5.0(1.6)	4.9(1.3)	5.0(1.2)
	MDGAN-v2	299.9(457)	300.4(457)	200.0(398)	4.8(3.0)	4.7(3.0)	5.5(2.6)
	UnrolledGAN	934.7(107)	874.1(290)	884.9(290)	0.72(0.51)	0.98(1.46)	0.90(1.4)
	VEEGAN	974.2(10.3)	687.9(447)	395.6(466)	0.33(0.05)	2.04(2.61)	3.52(2.64)
	GDPP	894.2(298)	897.1(299)	997.5(1.4)	0.92(1.92)	0.88(1.93)	**0.20**(0.02)
	PacGAN2	989.8(4.0)	993.3(4.8)	897.7(299)	0.33(0.02)	0.29(0.04)	0.87(1.94)
	BuresGAN (ours)	**993.5**(2.7)	**996.3**(1.6)	997.1(2.4)	0.29(0.02)	0.25(0.02)	0.23(0.01)

Table 4.1.3: Stacked MNIST experiment for an architecture with 4 conv. layers. All the models are trained for 25k iterations with a batch size of 64, a learning rate of 2×10^{-4} for Adam, and a normal latent distribution. The evaluation is over 10k samples and we report the average(std) over 10 runs.

		Nb modes(↑)	KL div.(↓)
4 conv. layers	GAN	21.6(25.8)	5.10(0.83)
	WGAN-GP	**999.7**(0.6)	**0.11**(0.006)
	MDGAN-v2	729.5(297.9)	1.76(1.65)
	UnrolledGAN	24.3(23.61)	4.96(0.68)
	VEEGAN	816.1(269.6)	1.33(1.46)
	GDPP	33.3(39.4)	4.92(0.80)
	PacGAN2	972.4(12.0)	0.45(0.06)
	BuresGAN (ours)	989.9(4.7)	0.38(0.06)

variability over the different experiments. MDGANv2, VEEGAN, GDPP and WGAN-GP often have an excellent single run performance. However, when increasing the number of discriminator layers, the training of these models has a tendency to collapse more often as indicated by the large standard deviation. Vanilla GAN is one of the best performing models in the variant with 3 layers.

We observe in Table 4.1.3 for the additional experiment that vanilla GAN and GDPP collapse for this architecture. WGAN-GP yields the best result and is

followed by BuresGAN. However, as indicated in Table 4.1.2, WGAN-GP is sensitive to the choice of architecture and hyperparameters and its training time is also longer as it can be seen from the corresponding timings table in SM. More generally, these results depend heavily on the precise architecture choice and to a lesser extent on the batch size. These experiments further confirm the finding that most GAN models, including the standard version, can learn all modes with careful and sufficient architecture tuning [141, 158]. Finally, it can be concluded that BuresGAN performs well for all settings, showing that it is robust when it comes to batch size and architecture.

Real images

Metrics. Image quality is assessed thanks to the Inception Score (IS), Fréchet Inception Distance (FID) and Sliced Wasserstein Distance (SWD). The latter was also used in [74] and [117] to evaluate image quality as well as mode-collapse. In a word, SWD evaluates the multiscale statistical similarity between distributions of local image patches drawn from Laplacian pyramids. A small Wasserstein distance indicates that the distribution of the patches is similar, thus real and fake images appear similar in both appearance and variation at this spatial resolution. The metrics are calculated based on 10k generated images for all the models.

CIFAR data sets. We evaluate the GANs on the $32\times32\times3$ CIFAR data sets, for which all models are trained for 100k iterations with a convolutional architecture. In Table 4.1.4, the best performance is observed for BuresGAN in terms of image quality, measured by FID and Inception Score, and in terms of mode collapse, measured by SWD. We also notice that UnrolledGAN, VEEGAN and WGAN-GP have difficulty converging to a satisfactory result for this architecture. This is in contrast to the 'simpler' synthetic data and the Stacked MNIST data set, where the models attain a performance comparable to BuresGAN. Also, for this architecture and number of training iterations, MDGAN did not converge to a meaningful result in our simulations. In [12], WGAN-GP achieves very good performance on CIFAR-10 with a ResNet architecture which is considerably more complicated than the DCGAN used here. Therefore, results with a Resnet architecture are reported in Section 4.1.4.

CIFAR-100 data set consists of 100 different classes and is, therefore, more diverse. Compared to the original CIFAR-10 data set, the performance of the studied GANs remains almost the same. An exception is vanilla GAN, which shows a higher presence of mode collapse as measured by SWD.

Table 4.1.4: Generation quality on CIFAR-10, CIFAR-100, and STL-10 with DCGAN architecture. For the CIFAR images, Average(std) over 10 runs and 100k iterations for each. For the STL-10 images, Average(std) over 5 runs and 150k iterations for each. For improving readability, the SWD score was multiplied by 100. The symbol '∼' indicates that no meaningful result could be obtained for these parameters.

	CIFAR-10			CIFAR-100			STL-10		
	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)
GAN	5.67(0.22)	59(8.5)	3.7(0.9)	5.2(1.1)	91.7(66)	7.8(4.9)	2.9(1.8)	237(54)	12.3(4.1)
WGAN-GP	2.01(0.47)	291(87)	8.3(1.9)	1.2(0.5)	283(113)	9.7(2.5)	∼	∼	∼
UnrolledGAN	3.1(0.6)	148(42)	9.0(5)	3.2(0.7)	172.9(40)	13.1(9.2)	∼	∼	∼
VEEGAN	2.5(0.6)	198(33.5)	12.0(3)	2.8(0.7)	177.2(27)	12.8(3.9)	∼	∼	∼
GDPP	5.76(0.27)	62.1(5.5)	4.1(1.1)	5.9(0.2)	65.0(8)	4.4(1.9)	3.3(2.2)	232(84)	8.2(4.0)
PacGAN2	5.51(0.18)	60.61(5.9)	2.95(1)	5.6(0.1)	59.9(5.2)	4.0(1.8)	4.7(1.5)	161(36)	8.1(4.3)
BuresGAN (ours)	**6.34**(0.17)	**43.7**(0.9)	**2.1**(0.6)	**6.5**(0.1)	**47.2**(1.2)	**2.1**(1.0)	**7.6**(0.3)	**109**(7)	**2.3**(0.3)



Figure 4.1.4: Images generated by BuresGAN with a ResNet architecture for CIFAR-10 (left) and STL-10 (right) data sets. The STL-10 samples are full-sized 96×96 images.

STL-10. The STL-10 data set includes higher resolution images of size $96 \times 96 \times 3$. The best-performing models from previous experiments are trained for 150k iterations. Samples of generated images from BuresGAN are given on Figure 4.1.4. The metrics are calculated based on 5k generated images for all the models. Compared to the previous data sets, GDPP and vanilla GAN are rarely able to generate high-quality images on the higher resolution STL-10 data set. Only BuresGANs are capable of consistently generating high-quality images as well as preventing mode collapse, for the same architecture.

Timings. The computing times for these data sets are in SM. For the same number of iterations, BuresGAN training time is comparable to WGAN-GP training for the simple data in Table 4.1.1. For more complicated architectures, BuresGAN scales better and the training time was observed to be significantly shorter with respect to WGAN-GP and several other methods.

4.1.4 High-quality generation using a ResNet architecture

Table 4.1.5: Best achieved IS and FID, using a ResNet architecture. Results with an asterisk are quoted from their respective papers (std in parenthesis). BuresGAN results were obtained after $300k$ iterations and averaged over 3 runs. The result indicated with † are taken from [247]. For all the methods, the STL-10 images are rescaled to $48 \times 48 \times 3$ in contrast with Table 4.1.4.

	CIFAR-10		STL-10	
	IS (↑)	FID (↓)	IS (↑)	FID (↓)
WGAN-GP ResNet [91]*	7.86(0.07)	18.8†	/	/
InfoMax-GAN [134]*	8.08(0.08)	17.14(0.20)	8.54(0.12)	37.49(0.05)
SN-GAN ResNet [162]*	8.22(0.05)	21.7(0.21)	9.10(0.04)	40.1(0.50)
ProgressiveGAN [117]*	8.80(0.05)	/	/	/
CR-GAN [258]*	8.4	14.56	/	/
NCSN [216]*	**8.87**(0.12)	25.32	/	/
Improving MMD GAN [241]*	8.29	16.21	9.34	37.63
WGAN-div [247]*	/	18.1†	/	/
BuresGAN ResNet (Ours)	8.81(0.08)	**12.91**(0.40)	**9.67**(0.19)	**31.42**(1.01)

As noted by [149], a fair comparison should involve GANs with the same architecture, and this is why we restricted in our section to a classical DCGAN architecture. It is natural to question the performance of BuresGAN with a ResNet architecture. Hence, we trained BuresGAN on the CIFAR-10 and STL-10 data sets by using the ResNet architecture taken from [91]. In this section, the STL-10 images are rescaled to a resolution of $48 \times 48 \times 3$ according to the procedure described in [162, 134, 241], so that the comparison of IS and FID scores with other works is meaningful. Note that BuresGAN has no parameters to tune, except for the hyperparameters of the optimizers.

The results are displayed in Table 4.1.5, where the scores of state-of-the-art unconditional GAN models with a ResNet architecture are also reported. In contrast with Section 4.1.3, we report here the best performance achieved at any time during the training, averaged over several runs. To the best of our

knowledge, our method achieves a new state-of-the-art inception score on STL-10 and is within a standard deviation of the state-of-the-art on CIFAR-10 using a ResNet architecture. The FID score achieved by BuresGAN is nonetheless smaller than the reported FID scores for GANs using a ResNet architecture. A visual inspection of the generated images in Figure 4.1.4 shows that the high inception score is warranted, the samples are clear, diverse, and often recognizable. BuresGAN also performs well on the full-sized STL-10 data set where an inception score of 11.11 ± 0.19 and an FID of 50.9 ± 0.13 is achieved (average and std over 3 runs).

4.2 Leverage score sampling for complete mode coverage in generative adversarial networks

4.2.1 Introduction

Complete mode coverage is a problem of generative models which has been clearly defined and studied in [263]. In layman's terms, a mode is defined as a local maximum of the data probability density. A closely related problem is mode collapse in GANs [88], which happens when a generative model is only capable of generating samples from a subset of all the modes. Multiple GAN variants have been proposed as a solution to this problem, however proposed solutions often assume that every mode has an (almost) equal probability of being sampled, which is often not the case in realistic datasets. Regularly, in critical applications, datasets contain a mixture of different subpopulations where the frequency of each subpopulation can be vastly different. The role of less abundant subpopulations in machine learning data has been discussed recently in [75]. Also, it is often common to presume that an algorithm does not know the abundance of subpopulations. It is however important that a machine learning model performs well on all subpopulations. A standard example is medical data where some rare diseases are less abundant than common diseases. To illustrate the approach presented in this section, a motivating example containing one majority mode and two minority modes is given in Figure 4.2.1. When sampling a mini-batch from the Probability Density Function (PDF) p, the side modes can be missed. We observe empirically that this is resolved by sampling from the Ridge leverage score (RLS) distribution (see Section 4.2.2), which has been extensively used in randomized linear algebra and kernel methods. Figure 4.2.1 shows that the samples from the minority modes have larger RLSs. Thus, when sampling from the RLS distribution, there is a higher probability of including the minority modes.

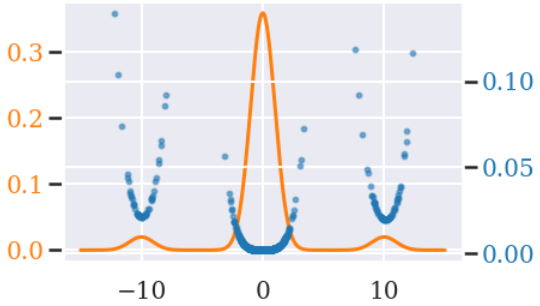


Figure 4.2.1: Probability Density Function (orange) and RLS of a sample of this PDF (blue). We take the motivating example from [263], which consists of a 1D target PDF p with 1 majority mode and 2 minority modes: $p = 0.9 \cdot \mathcal{N}(0,1) + 0.05 \cdot \mathcal{N}(10,1) + 0.05 \cdot \mathcal{N}(-10,1)$ given in orange. The RLS distribution is calculated using a Gaussian kernel with $\sigma = 3$ and $\gamma = 10^{-3}$. When sampling a mini-batch from the PDF p, the side modes can be missed. This is resolved by sampling from the RLS distribution.

This section is motivated by two situations where minority modes can occur: 1) the observed empirical distribution is different from the true distribution (biased data), and the data needs to be rebalanced. 2) The observed empirical distribution approximates the true distribution sufficiently well, but minority modes consist out of infrequent but very important points, e.g. rare diseases in a medical dataset.

Contribution. When training classical GANs, an empirical expectation of a loss $\mathbb{E}_{\boldsymbol{x}\sim p_d}[\mathcal{L}(\boldsymbol{x})$ is optimized in the context of a min-max problem. In this work, we propose a sampling procedure that promotes sampling out of minority modes by using ridge leverage scores. The common algorithmic procedure simulates the empirical distribution over the dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ by uniformly sampling over this set. We intentionally *bias* or *distort* this process by sampling $\boldsymbol{x}_i$ with probability $p(\boldsymbol{x}_i) \propto \ell_i$, where ℓ_i is the i-th ridge leverage score, defined in Section 4.2.2. Empirical evidence shows that our procedure rebalances the training distribution, as a result, the GAN model generates samples more uniformly over all modes. RLS sampling can easily be applied to any GAN to improve mode coverage. In particular, using our procedure in combination with a state-of-the-art method for complete mode coverage [263] shows a clear improvement. Finally, RLS sampling is combined with BuresGAN and a state-of-the-art StyleGAN2 with differentiable data augmentations [262], which in both cases improves mode coverage.

Related work. Several works discuss alternative sampling strategies in machine learning. In the context of risk-averse learning, the authors of [51] discuss an adaptive sampling algorithm that performs a stochastic optimization of the Conditional Value-at-Risk (CVaR) of a loss distribution. This strategy promotes models which do not only perform well on average but also on rare data points. In the context of generative models, AdaGAN [228] is a boosting approach to solve the missing mode problem, where at every step a new component is added into a mixture model by running the GAN training algorithm on a re-weighted sample. A supervised weighting strategy for GANs is proposed in [63]. In this section, we compare against two state-of-the-art GANs that combat mode collapse, PacGAN [141] and BuresGAN. PacGAN uses a procedure called packing. This modifies the discriminator to make decisions based on multiple samples from the same class, either real or artificially generated. In BuresGAN, an additional diversity metric in the form of the Bures distance between real and fake covariance matrices is added to the generator loss. Note that these methods tackle the traditional mode collapse problem, i.e., the data does not include minority modes. In [256, 257], it was shown that the convergence speed of stochastic gradient descent can be improved by actively selecting mini-batches using DPPs. In [215], coreset-selection is used to create mini-batches with a 'coverage' similar to that of the large batch – in particular, the small batch tries to 'cover' all the same modes as are covered in the large batch.

Before proceeding further, we discuss two main competitors more in-depth. The authors of [63] propose a solution to reduce selection bias in training data named Importance Weighted Generative Networks. A rescaling of the empirical data distribution is performed during training by employing a weighted Maximum Mean Discrepancy (MMD) loss such that the regions where the observed and the target distributions differ are penalized more. Each sample $i \in \{1, \dots, n\}$ is scaled by $1/M(\boldsymbol{x}_i)$, where M is the known or estimated Radon-Nykodym derivative between the target and observed distribution. A version of the vanilla GAN with importance weighting is introduced (IwGAN), as well as the weighting combined with MMDGAN (IwMmdGAN). Another approach to complete mode coverage by [263] and dubbed MwuGAN in this section, iteratively trains a mixture of generators. At each iteration, the sampling probability is pointwise normalized so that the probability to sample a missing mode is increased. Hence, this generates a sequence of generative models which constitutes the mixture. More precisely, a weight $w_i > 0$ is given for each $i \in \{1, \dots, n\}$ and initialized such that $w_i = p(\boldsymbol{x}_i)$ for some distribution[2] p. Next, a generative model is trained and the probability density $p_g(\boldsymbol{x}_i)$ of each $i \in \{1, \dots, n\}$ is computed.

[2]In [263], this initial distribution is uniform. We discuss in Section 4.2.3 a choice of weights based on RLSs and initialize MwuGAN with the normalized RLSs in (4.2.3).

If $p_g(\boldsymbol{x}_i) < \delta p(\boldsymbol{x}_i)$ for some threshold value $\delta \in (0,1)$, the weight is updated as follows: $w_i \leftarrow 2w_i$, otherwise the weight is not updated. The probability is then recalculated as follows: $p(\boldsymbol{x}_i) = w_i / \sum_j w_j$ for each $i \in \{1,\dots,n\}$. Another generative model is then trained by using $p(\boldsymbol{x}_i)$ and the procedure is repeated.

Classical approach. A GAN consists of a discriminator $D : \mathbb{R}^d \to \mathbb{R}$ and a generator $G : \mathbb{R}^\ell \to \mathbb{R}^d$ which are typically defined by neural networks, and parametrized by real vectors. The value $D(\boldsymbol{x})$ gives the probability that $\boldsymbol{x}$ comes from the empirical distribution, while the generator G maps a point $\boldsymbol{z}$ in the latent space $\mathbb{R}^\ell$ to a point in input space $\mathbb{R}^d$. A typical training scheme for a GAN consists in solving, in an alternating way, the following problems:

$$\begin{aligned} &\max_D \mathbb{E}_{\boldsymbol{x}\sim p_d}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_g}[\log(1 - D(\tilde{\boldsymbol{x}}))], \\ &\min_G -\mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_g}[\log(D(\tilde{\boldsymbol{x}}))], \end{aligned} \tag{4.2.1}$$

which include the vanilla GAN objective associated with the cross-entropy loss. In (4.2.1), the first expectation is over the empirical data distribution p_d and the second is over the generated data distribution p_g, implicitly given by the mapping by G of the latent prior distribution $\mathcal{N}(0, \mathbb{I}_\ell)$. The data distribution p_d is estimated using the empirical distribution over the training data $\hat{p}_d(\boldsymbol{x}) = \frac{1}{n}\sum_{\boldsymbol{x}_i\in\mathcal{D}} \delta(\boldsymbol{x} - \boldsymbol{x}_i)$ as follows: $\mathbb{E}_{p_d(\boldsymbol{x})}[\mathcal{L}(\boldsymbol{x}) \approx \mathbb{E}_{\hat{p}_d(\boldsymbol{x})}[\mathcal{L}(\boldsymbol{x}) = \frac{1}{n}\sum_{\boldsymbol{x}_i\in\mathcal{D}} \mathcal{L}(\boldsymbol{x}_i)$, where $\mathcal{L}$ is a general loss function. As noted by [233], positive weights w_i for $1 \le i \le n$ can be used to construct a weighted empirical distribution $\hat{p}_d^w(\boldsymbol{x}) = \sum_{\boldsymbol{x}_i\in\mathcal{D}} w_i \delta(\boldsymbol{x} - \boldsymbol{x}_i)$, then one can apply a weighting strategy to use samples distributed according to $\hat{p}(\boldsymbol{x})$ to estimate quantities with respect to $\hat{p}_d^w(\boldsymbol{x})$ as follows:

$$\mathbb{E}_{\hat{p}_d^w(\boldsymbol{x})}[\mathcal{L}(\boldsymbol{x})] = \mathbb{E}_{\hat{p}_d(\boldsymbol{x})}\left[\frac{\hat{p}_d^w(\boldsymbol{x})}{\hat{p}_d(\boldsymbol{x})}\mathcal{L}(\boldsymbol{x})\right] = \sum_{\boldsymbol{x}_i\in\mathcal{D}} w_i \mathcal{L}(\boldsymbol{x}_i). \tag{4.2.2}$$

A stochastic procedure is applied for minimizing the above expectation over $\hat{p}_d^w$. In this section, mini-batches are sampled according to the distribution $\hat{p}_d^w$ with w_i given by the normalized RLSs (4.2.3) for $1 \le i \le n$.

4.2.2 Sampling with ridge leverage scores

We propose to use a sampling procedure based on ridge leverage scores (RLSs) [9, 151]. RLSs correspond to the correlation between the singular vectors of a matrix and the canonical basis elements. The higher the score, the more *unique* the point. A sample from a minority mode would thus get a higher RLS. These

RLSs are used to bias the sampling, which in turn results in a more uniform sampling over all the modes, regardless of the original weight of the mode in the data distribution. Given a feature map $\varphi(\cdot)$, the corresponding kernel function is $K(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{y})$. Let the regularization parameter be $\gamma > 0$. Then, the γ-RLSs are defined for all $1 \leq i \leq n$ as:

$$\ell_i(\gamma) = \left(K(K + n\gamma\mathbb{I})^{-1}\right)_{ii} = \varphi(\boldsymbol{x}_i)^\top (C + n\gamma\mathbb{I})^{-1} \varphi(\boldsymbol{x}_i), \tag{4.2.3}$$

where $C = \sum_{i=1}^n \varphi(\boldsymbol{x}_i)\varphi(\boldsymbol{x}_i)^\top$ and $K_{ij} = \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x}_j)$ for $1 \leq i, j \leq n$. They have both a primal and a dual expression that can be leveraged when the size of the feature map or batch-size respectively are too large. When both the batch-size and feature map dimensions are large, one can rely on fast and reliable approximation algorithms with guarantees such as RRLS [165] and BLESS [200]. The role of $\gamma > 0$ is to filter the small eigenvalues of K in the spirit of Tikhonov regularization. RLSs induce the probability distribution: $p_i = \ell_i / \sum_{j=1}^n \ell_j$, for $1 \leq i \leq n$, which is used classically in randomized methods [9]. Figure 4.2.2 illustrates the interpretation of RLSs on two artificial datasets used in this section. The datasets consist of a mixture of Gaussians. In the Ring example, the first 4 modes are minority modes (starting on top and going further clockwise). In the Grid example, the first 10 modes are minority modes (starting left). Similar to the first illustration (see Figure 4.2.1), large RLSs are associated with minority modes. More information on the artificial datasets is given in Section 4.2.3.



Figure 4.2.2: RLS distribution using a Gaussian feature map with $\sigma = 0.15$ and regularization $\gamma = 10^{-3}$ on the unbalanced Ring (left) and Grid (right) data. The darker the shade, the higher the RLS. Dark modes correspond to minority modes.

RLS sampling has a rich history in kernel methods and randomized linear algebra but has not been used in the context of GANs. One of the key contributions of this section is to illustrate the use of RLSs in this setting. To do so, we propose the use of different feature maps so that RLS sampling can be used both for low dimensional and image data. In what follows, the feature map construction is first discussed. Next, two approximation schemes are introduced.

Choice of the feature map. Three choices of feature maps are considered in this section to compute leverage scores:

- Fixed implicit feature map. In low dimensional examples, the feature map can be chosen implicitly such that it corresponds to the Gaussian kernel: $\varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{y}) = \exp\left(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/\sigma^2\right)$, the bandwidth σ is a hyperparameter.
- Fixed explicit feature map. For image-based data, more advanced similarity metrics are necessary. Therefore, the next-to-last layer of a pre-trained classifier, e.g. the Inception network, is used to extract meaningful features. Note that the classifier does not need to be trained on the exact training dataset, but simply needs to extract useful features.
- Discriminator-based explicit feature map. The feature map can be obtained from the next-to-last layer of the discriminator. Let $D(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \varphi_D(\boldsymbol{x}))$, where $\boldsymbol{w} \in \mathbb{R}^m$ contains the last dense layer's weights and σ is the sigmoid function. This feature map is useful in situations where no prior knowledge is available about the dataset.

For a fixed feature map, the RLSs only need to be calculated once before training. The discriminator-based explicit feature map changes throughout the training. Therefore the RLSs are recalculated at every step. Nonetheless, due to approximation schemes which are discussed hereafter, the computational cost stays low. The full algorithm is given in Appendix D.5.

Approximation schemes

Current day models are high dimensional, e.g., a DCGAN yields a feature space $\mathbb{R}^m$ of high dimension $m = 10^3$. Moreover, the size of datasets is commonly thousands up to millions of images. Therefore, two approximation schemes are proposed to speed up the computation of RLSs when using explicit feature maps:

- For the discriminator-based explicit feature map, we propose a two-stage sampling procedure in combination with a Gaussian sketch to reduce the dimension of high dimensional feature maps.
- For the fixed explicit feature map, the well-known UMAP is used to reduce the dimension [156].

Two-stage sampling procedure. For the explicit discriminator-based feature map, φ_D has to be re-calculated at each training step. To speed up the sampling procedure, we propose a sampling procedure in two stages. First, a subset of the data is uniformly sampled, e.g. equal to 20 times the desired batch size. Afterward, the RLSs are calculated only for the uniformly sampled subset, which are then used to sample the final batch used for training. This two-stage sampling procedure is similar to the core-set selection used in smallGAN [215]. A first difference is that core-sets are selected by combining a Gaussian sketch and a greedy selection in [215], while we use a randomized approach. Second, in this reference, cores-sets are used to reduce the batch size to improve scalability. In contrast, RLS sampling is used here to bias the empirical distribution.

Sketching the discriminator feature map. Gaussian sketching is a commonly used method to reduce data dimension and was also used in [215] and [253] to reduce the dimension of large neural nets. Let S be a sketching matrix of size $m \times k$ such that $S = A/\sqrt{k}$ with A a matrix with i.i.d. zero-mean standard normal entries. Consider the following random projection: let a batch be $\{i_1, \ldots, i_b\} \subset \{1, \ldots, n\}$. A random projection of this batch in feature space is then defined as follows:

$$\varphi(\boldsymbol{x}_{i_\ell}) = S^\top \varphi_D(\boldsymbol{x}_{i_\ell}) \in \mathbb{R}^k, \tag{4.2.4}$$

for all $\ell \in \{1, \ldots, b\}$. This random projection preserves approximately (squared) pairwise distances in the dataset and is motivated by an isometric embedding result in the spirit of Johnson-Lindenstrauss lemma. Let $0 < \epsilon < 1$ and any integer $b > 0$. Let k be an integer such that $k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \log b$. Then, for any set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_b\}$ in $\mathbb{R}^m$ there is a map $f : \mathbb{R}^m \to \mathbb{R}^k$ such that for any $\ell, \ell' \in \{1, \ldots, b\}$ we have

$$(1 - \epsilon)\|\boldsymbol{x}_\ell - \boldsymbol{x}_{\ell'}\|_2^2 \leq \|f(\boldsymbol{x}_\ell) - f(\boldsymbol{x}_{\ell'})\|_2^2 \leq (1 + \epsilon)\|\boldsymbol{x}_\ell - \boldsymbol{x}_{\ell'}\|_2^2.$$

The idea of this work is to consider the set of points given by the batch in the discriminator feature space $\boldsymbol{x}_\ell = \varphi_D(\boldsymbol{x}_{i_\ell})$ for $1 \leq \ell \leq b$. It is proved in [52] that f exists and can be obtained with high probability with a random projection of the form (4.2.4). For a more detailed discussion, we refer to [173].

Dimensionality reduction of the fixed explicit feature map by UMAP. The Gaussian sketch is a simple and fast method to reduce the dimension of the feature map. This makes it a perfect candidate to reduce the dimension of the proposed discriminator feature map, which has to be recalculated at every iteration. Unfortunately, this speed comes at a price, namely, the gaussian sketch is deemed too simple to reduce the dimension of highly complex models

like the Inception network. Therefore, UMAP is proposed [156]. This non-linear dimensionality reduction technique can extract more meaningful features. UMAP is considerably slower than the Gaussian sketch, therefore the use of UMAP is only advised for a fixed feature map like a pre-trained classifier or the Inception network, where the RLSs are only calculated once before training.

4.2.3 Numerical experiments

The training procedure is evaluated on several synthetic and real datasets, where we artificially introduce minority modes. The GANs are evaluated by analyzing the distribution of the generated samples. Ideally, the models should generate samples from every mode as uniformly as possible. A re-balancing effect should be visible. The proposed methods are compared with vanilla GAN, PacGAN, MwuGAN, BuresGAN, IwGAN, and IwMmdGAN. In particular, MwuGAN outperforms AdAGAN [228] on complete mode coverage problems [263].

Overview of proposed methods. The RLS sampling procedure can easily be integrated into any GAN architecture or model. In this spirit, we used RLS sampling with a classical vanilla GAN (**RLS GAN**) and BuresGAN (**RLS BuresGAN**). The classical BuresGAN has been shown to outperform competitors in mode collapse problems. We noticed empirically that RLS BuresGAN outperformed RLS GAN on the synthetic data (the comparison is shown in Appendix D.3). Therefore we only continue with RLS BuresGAN in the rest of the experiments. Likewise, RLS sampling is combined with MwuGAN, which is considered state-of-the-art in complete mode coverage. The method, called **RLS MwuGAN**, uses RLSs as initial starting weights to sample as opposed to uniform weights. Besides the initialization, the method remains unchanged. The number of generators in the mixture is always displayed in brackets. Unless specified otherwise, the models are trained for 30k iterations with a batch size of 64, by using the Adam [121] optimizer. Unless specified otherwise, we report the means and standard deviations for 10 runs. The largest mean is depicted in black, a * represents significance using a one-tailed Welch's t-test between the best performing proposed model and best performing competitor at a 0.05 confidence level. Further information about the used architectures, hyperparameters, and timings are given in Appendix D.5.

Synthetic data

Unbalanced versions of two classical synthetic datasets are generated: an unbalanced ring with 4 minority modes (Ring) and an unbalanced grid (Grid) with 10 minority modes (see Figure 4.2.2). Ring is a mixture of

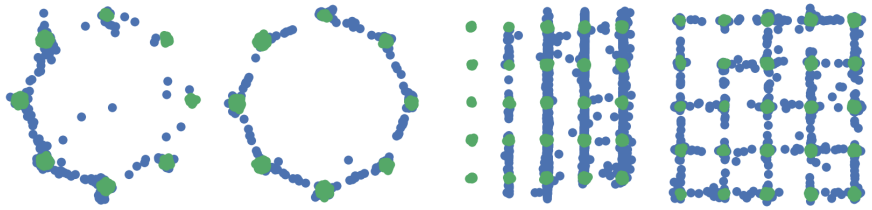


Figure 4.2.3: Visualization of the generation quality on RING and GRID. Each column shows 2.5k samples from the trained generator in blue and 2.5k samples from the true distribution in green. The vanilla GAN (first and third) does not cover the minority modes. This is not the case for the RLS BuresGAN Discr. (second and fourth).

eight two-dimensional isotropic Gaussians in the plane with means $2.5 \times (\cos((2\pi/8)i), \sin((2\pi/8)i))$ and std 0.05 for $i \in \{1, \dots, 8\}$. The probability of sampling from the first 4 consecutive Gaussians is only 0.05 times the probability of sampling from the last 4 modes. GRID is a mixture of 25 two-dimensional isotropic normals with standard deviation 0.05 and with means on a square grid with spacing 2. The first rectangular blocks of 2×5 adjacent modes are depleted with a factor 0.05.

Table 4.2.1: Experiments on the synthetic datasets RING and GRID. Two RLS BuresGAN are considered: RLSs calculated with the Gaussian kernel (Gauss.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using an implicit feature map associated with the Gaussian kernel.

	Ring with 8 modes		Grid with 25 modes	
	Nb modes (↑)	% in 3σ (↑)	Nb modes (↑)	% in 3σ (↑)
GAN	5.0(1.1)	**0.92**(0.02)*	8.3(3.4)	0.29(0.3)
PacGAN2	5.4(1.4)	**0.92**(0.03)	10.3(2.6)	0.13(0.02)
BuresGAN	5.8(1.4)	0.76(0.27)	16.7(0.9)	**0.82**(0.01)*
RLS GAN Gauss.	7.4(0.9)	0.86(0.05)	13.8(7.4)	0.51(0.32)
RLS GAN Discr.	7.6(0.8)	0.90(0.02)	20.4(2.6)	0.81(0.03)
IwGAN	**8**(0)	0.85(0.08)	13.4(5.7)	0.29(0.25)
IwMmdGAN	**8**(0)	0.84(0.02)	1.7(5.1)	0.03(0.05)
MwuGAN (15)	7.9(0.3)	0.86(0.02)	15.2(1.7)	0.47(0.11)
RLS MwuGAN (15) (ours)	**8**(0)	0.84(0.06)	22.3(1.9)	0.60(0.1)
RLS BuresGAN Gauss. (ours)	**8**(0)	0.90(0.02)	24.0(1.5)	0.76(0.11)
RLS BuresGAN Discr. (ours)	**8**(0)	0.90(0.02)	**24.4**(0.92)*	0.78(0.06)

Evaluation. The evaluation is done by sampling 10k points from the generator network. High-quality samples are within 3 standard deviations of the nearest mode. A mode is covered if there are at least 50 generated samples within 3 standard deviations of the center of the mode. The knowledge of the full Radon-Nikodym derivative M is given to IwMmdGAN and IwGAN by dividing the true probability of each sample (a Gaussian mixture with equal weights) by the adapted Gaussian mixture containing several minority modes. The results of the experiments are given in Table 4.2.1. For RLS sampling, we use a Gaussian kernel with bandwidth $\sigma = 0.15$, and the discriminator network as feature extractor, both with regularization parameter $\gamma = 10^{-3}$. The models are trained using a fully connected architecture (see Appendix D.5). As the models are rather simple, no dimensionality reduction is needed.

Generated samples from models trained with and without RLS sampling are displayed in Figure 4.2.3. One can clearly see that training a GAN with uniform sampling results in missing the first 4 minority modes. This is solved by using RLS sampling and can be interpreted by comparing the two sampling distributions on Figure 4.2.4 (uniform) and Figure 4.2.2 (RLS). The RLSs are larger for samples in minority modes, which results in a more uniform mini-batch over all modes. Note that the RLS sampling procedure, given the feature map, is completely unsupervised and has no knowledge of the desired unbiased distribution. The evaluation metrics in Table 4.2.1 confirm our suspicions. Only methods designed for complete mode coverage can recover (almost) all modes for the Ring dataset. For the unbalanced Grid, only the proposed method has an acceptable performance. Our method even outperforms multiple generator architectures like MwuGAN and RLS MwuGAN, which are considerably more costly to train. Moreover, IwGAN, with full knowledge of M, is not capable of consistently capturing all modes. This was pointed out by the authors in [63]: the method may still experience high variance if it rarely sees data points from a class it wants to boost.

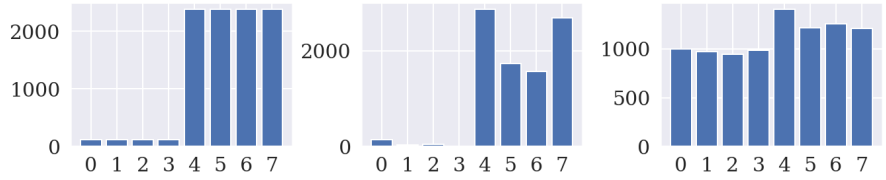


Figure 4.2.4: Ring. Number of training samples in each mode for the Ring dataset (left) Generated samples in each mode by a vanilla GAN (middle). Generated samples by RLS BuresGAN Discr. (right). A rebalancing effect is visible.

Unbalanced MNIST

For this experiment, we create two unbalanced datasets out of MNIST. The first modified dataset, named UNBALANCED 012-MNIST, consists of only the digits $0, 1$ and 2. The class 2 is depleted so that the probability of sampling 2 is only 0.05 times the probability of sampling from the digit 0 or 1. This is a similar experiment as in [63], although we increase here the difficulty by making the minority mode even more outspoken. The second dataset, named UNBALANCED MNIST, consists of all digits. The classes 0,1,2,3, and 4 are all depleted so that the probability of sampling out of the minority classes is only 0.05 times the probability of sampling from the majority digits. For these experiments, we use a DCGAN architecture. The following metrics are used for performance evaluation: the number of generated digits in each mode, which measures mode coverage, and the KL divergence [159] between the classified labels of the generated samples and a balanced label distribution, which measures sample quality. The mode of each generated image is identified by using a MNIST classifier which is trained up to 98.43% accuracy (see Appendix D.5). The metrics are calculated based on 10k generated images for all the models. For the RLS computation, we use both the discriminator with a Gaussian sketch and the next-to-last layer of the pre-trained classifier with a UMAP dimensionality reduction as a feature map. For the UNBALANCED 012-MNIST, both feature maps are reduced to $k = 25$ and the regularization parameter is $\gamma = 10^{-4}$. In the UNBALANCED MNIST, we take $k = 10$ and $\gamma = 10^{-4}$. An ablation study over different k and γ is given in Appendix D.3. We also compare the performance of the classical MwuGAN, initialized with uniform weights, with RLS MwuGAN where the weights are initialized by the RLSs calculated using the fixed explicit feature map with the same parameters mentioned above. Both methods contain a mixture of 15 GANs, the experiments are repeated 3 times for MwuGAN variants. In our simulations, IwMmdGAN could not be trained successfully

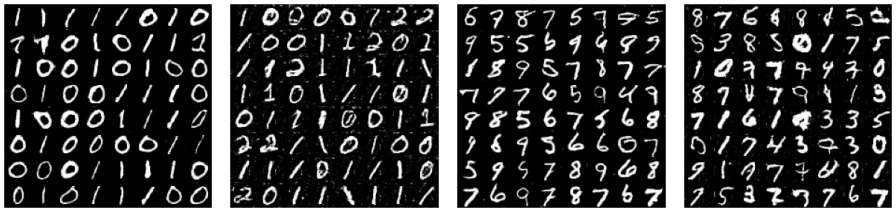


Figure 4.2.5: Generated images from UNBALANCED 012-MNIST by a vanilla GAN (first), by RLS BuresGAN Discr. (second) and generated images from UNBALANCED MNIST by a vanilla GAN (third), by RLS BuresGAN Class. (fourth). The minority digits are generated more frequently in the proposed methods that include RLS sampling.

with a DCGAN architecture. The Radon-Nikodym derivative M, which is used by IwMmdGAN and IwGAN, is defined as follows: $M_i = 1$ for digits 0 and 1 and $M_i = 0.05$ for digits 2, analogous for the UNBALANCEDMNIST dataset. Only the proposed models trained with RLS sampling are capable of covering all modes consistently. The diversity of images generated by RLS BuresGAN can be visualized in Figure 4.2.5 where digits from minority modes appear more frequently. A quantitative analysis of mode coverage and sample quality is reported in Tables 4.2.2 and 4.2.3. In the UNBALANCED 012-MNIST dataset, there is a clear advantage in using RLS sampling with BuresGAN since mode coverage and the KL divergence are improved compared to the other methods. The second best method is RLS MwuGAN which outperforms RLS with uniform starting weights in KL. For the more difficult UNBALANCED MNIST dataset, using the fixed explicit feature map to calculate the RLSs clearly outperforms other methods.

Table 4.2.2: Experiments on the UNBALANCED 012-MNIST dataset. Two RLS BuresGAN are considered: RLSs calculated with an explicit feature map obtained from a pre-trained classifier (Class.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using the explicit feature maps obtained from a pre-trained classifier. Minority modes are highlighted in black in the first row.

	Mode 1	Mode 2	**Mode 3**	KL
GAN	4381(172)	5412(179)	129(36)	0.31(0.01)
PacGAN2	4492(237)	5328(242)	123(29)	0.32(0.01)
BuresGAN	4586(287)	5190(292)	142(19)	0.30(0.01)
IwGAN	4368(295)	5414(287)	147(32)	0.32(0.01)
IwMmdGAN	34(12)	0(0)	69(12)	0.56(0.10)
MwuGAN (15)	4886(473)	4865(466)	176(14)	0.31(0.01)
RLS MwuGAN (15) (ours)	3982(218)	4666(164)	870(65)	0.14(0.01)
RLS BuresGAN Class. (ours)	3414(161)	4862(134)	1461(183)	**0.08**(0.01)*
RLS BuresGAN Discr. (ours)	5748(172)	2416(268)	**1566**(293)*	0.16(0.02)

Unbalanced CIFAR10

We conclude this section with an experiment on colored images, namely the CIFAR10 dataset. This highly diverse dataset contains 32×32 color images from 10 different classes. We consider two unbalanced variations. The first modified

Table 4.2.3: Experiments on the UNBALANCED MNIST dataset. Two RLS BuresGAN variants are considered: RLSs calculated with an explicit feature map obtained from a pre-trained classifier (Class.) and the next-to-last layer of the discriminator (Discr.). RLS MwuGAN is initialized with RLSs using the explicit feature maps obtained from a pre-trained classifier. Only the number of samples in the minority modes are visualized. The number of samples in the remaining modes are given in Appendix D.2

	Mode 1	**Mode 2**	**Mode 3**	**Mode 4**	**Mode 5**	KL
GAN	123(22)	137(103)	81(31)	161(97)	161(23)	0.48(0.02)
PacGAN2	109(29)	142(70)	89(23)	147(100)	152(40)	0.48(0.02)
BuresGAN	126(31)	157(97)	108(30)	153(62)	147(26)	0.46(0.02)
IwGAN	117(33)	139(33)	97(31)	212(69)	154(34)	0.46(0.02)
IwMmdGAN	1(1)	0(0)	23(16)	**1140**(367)*	3(3)	1.92(0.1)
MwuGAN (15)	144(29)	113(28)	146(13)	172(18)	167(28)	0.46(0.02)
RLS MwuGAN (15) (ours)	336(47)	283(32)	191(23)	381(38)	276(33)	0.30(0.02)
RLS BuresGAN Class. (ours)	**875**(112)*	**663**(122)*	**360**(198)*	831(59)	**615**(82)*	**0.09**(0.01)*
RLS BuresGAN Discr. (ours)	235(62)	183(141)	264(44)	255(109)	219(54)	0.37(0.02)

dataset, named UNBALANCED 06-CIFAR10, consists of only the classes 0 and 6 or images of airplanes and frogs respectively. The class 0 is depleted so that the probability of sampling an airplane is only 0.05 times the probability of sampling an image from the class frogs. The second dataset, named UNBALANCED 016-CIFAR10, consists of the classes 0,1 and 6. Compared to the previous dataset, we add images from the class automobile. Now, the class 6 consisting of frogs is depleted with a factor 0.05. We show the improvement of RLS sampling in a StyleGAN2 with differentiable data augmentation (StyleGAN2 + Aug.) [262] (the code is taken from `https://github.com/mit-han-lab/data-efficient-gans`). By clever use of various types of differentiable augmentations on both real and fake samples, the GAN can match the top performance on CIFAR10 with only 20% training data and is considered state-of-the-art. The StyleGAN2 models are trained for 156k iterations with a mini-batch size of 32 using 'color, translation, and cutout' augmentations, which is suggested by the authors when only part of the CIFAR10 dataset is used. All the other parameters remained the same, only the sampling strategy is changed to RLS sampling in RLS StyleGAN2 + Aug. For the RLS computation, we use the discriminator feature map with Gaussian sketching and a fixed explicit feature map given by the next-to-last layer of the Inception network where UMAP is used to reduce the dimension. For both the RLSs, the dimension is reduced to $k = 25$ and the regularization parameter is $\gamma = 10^{-4}$. The performance is assessed using 10k generated samples at the end of training by the Inception Score (IS) and the Fréchet inception distance (FID) between the generated fake dataset and

the balanced dataset. Mode coverage is evaluated by the number of generated

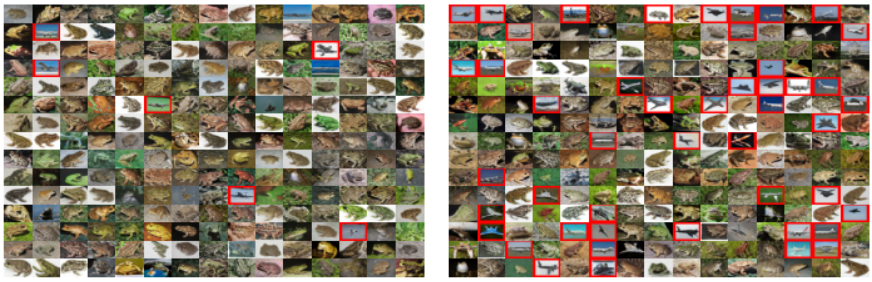


Figure 4.2.6: Generated images from 06-CIFAR10 by a StyleGAN2 + Aug. (left) and by RLS StyleGAN2 + Aug. (right). Including RLS sampling promotes sampling from the minority class. Generated samples classified as planes are marked by a red border.

samples in each class. The class of a generated sample is evaluated by a trained CIFAR10 classifier using a resnet56 type architecture [96] which is trained up to 93.77% accuracy[3]. The results of the experiments are given in Table 4.2.4, examples of generated images are given in Figure 4.2.6. Including RLS sampling clearly improves the performance in unbalanced datasets, this is especially the case for the fixed feature map given by the Inception network. The minority mode is oversampled by approximately a factor 10 or even 100 in the case of the unbalanced 06-CIFAR10 and 016-CIFAR10 datasets respectively. Both the IS and FID also improve significantly. Note that the maximum achievable performance for IS and FID is lower when only a subset of classes is included, as pointed out by [203].

Table 4.2.4: Experiments on the UNBALANCED 06-CIFAR10 and UNBALANCED 016-CIFAR10 dataset. Including RLS sampling in the StyleGAN2 + Aug. clearly improves the performance. Minority modes are highlighted in black in the second row.

	06-CIFAR10				016-CIFAR10				
	Mode 1	Mode 2	IS (↑)	FID (↓)	Mode 1	Mode 2	**Mode 3**	IS (↑)	FID (↓)
StyleGAN2 + Aug.	261	9500	4.8	67.5	4526	5206	18	4.3	48.8
RLS StyleGAN2 + Aug. Disc. (ours)	994	8659	5.7	46.4	4449	5132	139	4.6	44.4
RLS StyleGAN2 + Aug. Class. (ours)	**2438**	7212	**6.2**	**31.3**	4156	4393	**1155**	**5.7**	**27.2**

[3]Classifier is available at `https://github.com/gahaalt/ResNets-in-tensorflow2`

4.3 Conclusion

In this chapter, we analyzed the role of diversity as a solution to mode collapse in GANs. In Section 4.1, we proposed matching the real batch diversity to the fake batch diversity using the Bures distance between the covariance matrices in feature space. Our experiments showed that BuresGAN is capable of reducing mode collapse and achieve a clear improvement of sample quality without parameter tuning or the need for regularization such as a gradient penalty. In Section 4.2, BuresGAN is extended with ridge leverage score sampling. RLS sampling is easy to integrate into any GAN model with only minimal changes to the training procedure. Three feature maps have been discussed. An implicit feature map performs well for low-dimensional data. A fixed explicit feature map, such as a pre-trained classifier, achieves good results in high-dimensional cases. Lastly, the discriminator can be used as a feature map when no prior knowledge exists about the data. Two approximation methods for the explicit feature maps are also discussed: dimensionality reduction of explicit feature maps and a two-stage sampling procedure to efficiently speed up online RLS computation. It is shown that including this diversity sampling procedure significantly improves mode coverage when minority modes are present in the data.

Chapter 5

Disentanglement and restricted kernel machines

This chapter comprises previously published and submitted articles including:

- Joachim Schreurs[1] and Johan A. K. Suykens. Generative Kernel PCA. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 129–134, 2018
- David Winant, Joachim Schreurs[2], and Johan AK Suykens. Latent space exploration using generative kernel PCA. In *Proc. of Bogaerts B. et al. (eds) Artificial Intelligence and Machine Learning. Communications in Computer and Information Science, vol 1196. Springer, Cham. (BNAIC 2019, Benelearn 2019)*, pages 70–82. Springer, 2019
- Arun Pandey, Joachim Schreurs[2], and Johan AK Suykens. Generative restricted kernel machines: A framework for multi-view generation and disentangled feature learning. *Neural Networks*, 135:177–191, 2021
- Arun Pandey, Michael Fanuel, Joachim Schreurs[2], and Johan AK Suykens. Disentangled representation learning and generation with manifold optimization. *arXiv preprint arXiv:2006.07046*, 2020

[1]The candidate performed the research and wrote the article under the guidance of the coauthors.

[2]The candidate helped the first author with the research, methodology, and article preparation.

5.1 Latent space exploration using generative kernel PCA*

5.1.1 Introduction

Latent spaces provide a representation of data by embedding the data into an underlying vector space. Exploring these spaces allows for deeper insights into the structure of the data distribution, as well as understanding relationships between data points. Latent spaces are used for various purposes like latent space cartography [144], object shape generation [248], or style-based generation [118]. In this section, the focus will be on how the synthesis of new data with generative methods can help with understanding the latent features extracted from a dataset. An example of a real-world application of latent spaces using VAEs is shown in [242], where deep convolutional VAEs are used to extract a biologically meaningful latent space from a cancer transcriptomes dataset. This latent space is used to explore hypothetical gene expression profiles of tumors and their reaction to possible treatments. Similarly, disentangled variational autoencoders have been used to find an interpretable and explainable representation of ECG signals [236]. Latent space exploration is also used for interpreting GANs, where interpolation between different images allows for the interpretation of the different features captured by the latent space, such as windows and curtains when working with datasets of bedroom images [184].

In [224], kernel PCA was cast within the framework of Restricted Kernel Machines (RKMs) which allows for an interpretation in terms of hidden and visible units similar to a type of generative neural network known as Restricted Boltzmann Machines (RBMs) [100]. This connection between kernel PCA and RBMs is used to explore a generative mechanism for the kernel PCA. This mechanism is then used to explore the latent feature space extracted by kernel PCA, to interpret the components. This has led to the development of a Matlab tool that can be used to visualize the latent space of the kernel PCA method along its principal components. The use of the tool is demonstrated on three different datasets: the MNIST digits dataset, the Yale Face dataset, and the MIT-BIH Arrhythmia dataset. As a final illustration, feature space exploration is used in the context of novelty detection [103], where the latent space around novel patterns in the data is explored. This helps the interpretation of why certain points are considered novel.

In Section 5.1.2, the generative kernel PCA is introduced. In Section 5.1.3, latent feature space exploration is demonstrated. Subsequently, we will illustrate

*This section combines the works [210] and [245].

how latent feature space exploration can help in interpreting novelty detection in Section 5.1.4.

5.1.2 Kernel PCA in the RKM framework

In this section, the generative kernel PCA formulation is deduced. We start from the RKM representation of kernel PCA, that gives an upper bound on the original kernel PCA objective function. Given the training data $\{\boldsymbol{v}_i\}_{i=1}^N$:

$$\bar{J}_{\text{train}}(\boldsymbol{h}_i, U) = -\sum_{i=1}^{N} \boldsymbol{v}_i^\top U \boldsymbol{h}_i + \frac{\lambda}{2}\sum_{i=1}^{N} \boldsymbol{h}_i^\top \boldsymbol{h}_i + \frac{\eta}{2}\text{Tr}(U^\top U),$$

where $\boldsymbol{v}_i \in \mathbb{R}^d$ represents the visible unit and $\boldsymbol{h}_i \in \mathbb{R}^s$ the corresponding hidden unit for the ith datapoint, $U \in \mathbb{R}^{d\times s}$ the interaction matrix. Similar to RBMs, the input patterns are clamped to the visible units in order to learn the hidden units and interaction matrix. Stationary points of the training function $\bar{J}_{\text{train}}(\boldsymbol{h}_i, U)$ are given by:

$$\frac{\partial \bar{J}_{\text{train}}}{\partial \boldsymbol{h}_i} = 0 \Rightarrow U^\top \boldsymbol{v}_i = \lambda \boldsymbol{h}_i, \ \forall i \tag{5.1.1}$$

$$\frac{\partial \bar{J}_{\text{train}}}{\partial U} = 0 \Rightarrow U = \frac{1}{\eta}\sum_{i=1}^{N} \boldsymbol{v}_i \boldsymbol{h}_i^{\text{T}}. \tag{5.1.2}$$

Elimination of U results in the following eigenvalue problem, which corresponds to the original linear kernel PCA formulation [207]:

$$\frac{1}{\eta} K H^{\text{T}} = H^{\text{T}} \Delta,$$

where $H = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N] \in \mathbb{R}^{s\times N}$, $\Delta = \text{diag}\{\lambda_1, \ldots, \lambda_s\}$ with $s \leq N$ the number of selected components and $K_{ij} = \boldsymbol{v}_i^\top \boldsymbol{v}_j$ the kernel matrix elements. This can easily be extended to the nonlinear case by using the feature map and kernel trick by replacing $\boldsymbol{v}_i$ by $\varphi(\boldsymbol{v}_i)$ [224].

After training the model, we should be able to re-generate the visible units of the training dataset. The hidden units $\boldsymbol{h}_i$ and interaction matrix U are assumed to be known from training the model and correspond to equations (5.1.1) and (5.1.2). We propose the following generating objective function, by introducing a regularization term on the visible units:

$$\bar{J}_{\text{gen}}(\boldsymbol{v}_i) = -\sum_{i=1}^{N} \boldsymbol{v}_i^\top U \boldsymbol{h}_i + \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{v}_i^\top \boldsymbol{v}_i.$$

Stationary points of the generating objective function $\bar{J}_{\text{gen}}(\boldsymbol{v}_i)$ are given by:

$$\frac{\partial \bar{J}_{\text{gen}}}{\partial \boldsymbol{v}_i} = 0 \Rightarrow U\boldsymbol{h}_i = \boldsymbol{v}_i, \ \forall i.$$

One can easily see that filling in the hidden features $\boldsymbol{h}_i$ and U of the training phase, results in the original visible units $\boldsymbol{v}_i$.

A clear link with RBMs is visible [100, 101]. Similar to RBMs, there first occurs a training phase to find the hidden units, weights, and biases. Using the conditional distributions $p(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{\theta})$ and $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{\theta})$, the contrastive divergence algorithm is used to optimize the model parameters $\boldsymbol{\theta}$ [76]. The algorithm makes use of Gibbs sampling inside a gradient descent procedure to compute weight updates. After the RBM is trained, the model can be used to generate new samples. Given a visible unit $\boldsymbol{v}$, the model returns a hidden unit h and vice-versa. This mechanism is made possible by the energy function of the RBM [100, 101]:

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\boldsymbol{v}^\top U\boldsymbol{h} - c^\top \boldsymbol{v} - a^\top \boldsymbol{h},$$

with model parameters $\boldsymbol{\theta} = \{U, c, a\}$. The same property is present in the RKM objective function, which is a combination of $\bar{J}_{\text{train}}$ and $\bar{J}_{\text{gen}}$:

$$\bar{J}(\boldsymbol{v}, \boldsymbol{h}, U) = -\boldsymbol{v}^\top U\boldsymbol{h} + \frac{\lambda}{2}\boldsymbol{h}^\top \boldsymbol{h} + \frac{1}{2}\boldsymbol{v}^\top \boldsymbol{v} + \frac{\eta}{2}\text{Tr}(U^\top U), \tag{5.1.3}$$

which can be seen as a non-probabilistic variant of the RBM energy function. The training phase however consists of solving an eigenvalue problem. For generating the visible units, a matrix multiplication is needed in the RKM case.

Generative kernel PCA can be used to generate new data. Instead of using the hidden units of the training set, one could use a new hidden unit $\boldsymbol{h}^\star$. Similar to RBMs, a new hidden unit is clamped to the model. We propose generating new hidden units by fitting a normal distribution through the trained hidden units, with afterwards sampling from this distribution $p(\boldsymbol{h})$. As shown by Suykens et al. [223], kernel PCA corresponds to a one-class LS-SVM problem with zero target value around which one maximizes the variance. This property results in the hidden variables most typically having a normal distribution around zero (however for generating new hidden units other distributions are possible). The optimization problem, where U is obtained by the training phase in equation (5.1.2) and $\boldsymbol{h}^\star$ is sampled from a normal distribution, corresponds to:

$$\bar{J}_{\text{gen}}(\boldsymbol{v}^\star) = -\boldsymbol{v}^{\star\top} U\boldsymbol{h}^\star + \frac{1}{2}\boldsymbol{v}^{\star\top}\boldsymbol{v}^\star,$$

with $\boldsymbol{v}^\star$ generated by equation:

$$\boldsymbol{v}^\star = U\boldsymbol{h}^\star. \tag{5.1.4}$$

Dimensionality reduction and denoising

Linear case. In the linear case, the visible units are equal to the training points $\boldsymbol{v}_i = \boldsymbol{x}_i$. When using a subset of the trained hidden features $H \in \mathbb{R}^{s \times N}$ and trained interaction matrix U (see equations (5.1.1) and (5.1.2)) to re-generate the original dataset:

$$\hat{X} = UH = (\frac{1}{\eta} \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{h}_i^{\mathrm{T}})H = \frac{1}{\eta} X H^{\mathrm{T}} H, \tag{5.1.5}$$

with training dataset $X \in \mathbb{R}^{d \times N}$. This corresponds to minimizing the reconstruction error $\|X - \hat{X}\|^2$. The above equation is also equal to reconstruction or denoising using linear PCA [116].

Nonlinear case. In the nonlinear case, the visible units are equal to the feature map of the data points $\boldsymbol{v}_i = \varphi(\boldsymbol{x}_i)$ where $\varphi(\boldsymbol{x}_i) : \mathbb{R}^d \to \mathbb{R}^{n_f}$ is assumed to be a centered feature map [207]. A new datapoint $\boldsymbol{x}^\star$ is generated using the known corresponding hidden unit $\boldsymbol{h}^\star$. The generative equation (5.1.4) becomes:

$$\varphi(\boldsymbol{x}^\star) = U\boldsymbol{h}^\star = (\frac{1}{\eta} \sum_{i=1}^{N} \varphi(\boldsymbol{x}_i) \boldsymbol{h}_i^{\mathrm{T}})\boldsymbol{h}^\star,$$

where U is the trained interaction matrix of equation (5.1.2) and $\boldsymbol{h}_i$ the trained hidden unit of equation (5.1.1). However the above equation requires $\varphi(\boldsymbol{x}_i)$ in its explicit form. Finding the value original datapoint based on the mapping in the feature space is known as the pre-image problem [104].

To solve this problem, we propose to multiply both sides of the equation with the feature map of every training-point $\varphi(\boldsymbol{x}_j)$: $(\varphi(\boldsymbol{x}_j) \cdot \varphi(\boldsymbol{x}^\star)) = \frac{1}{\eta}(\varphi(\boldsymbol{x}_j) \cdot \sum_{i=1}^{N} \varphi(\boldsymbol{x}_i)\boldsymbol{h}_i^{\mathrm{T}})\boldsymbol{h}^\star$, where $j = 1, \ldots, N$. This results in the following equation:

$$K(\boldsymbol{x}_j, \boldsymbol{x}^\star) = \frac{1}{\eta}(\sum_{i=1}^{N} K(\boldsymbol{x}_j, \boldsymbol{x}_i)\boldsymbol{h}_i^{\mathrm{T}})\boldsymbol{h}^\star, \tag{5.1.6}$$

where $K(\boldsymbol{x}_j, \boldsymbol{x}^\star) = (\varphi(\boldsymbol{x}_j) \cdot \varphi(\boldsymbol{x}^\star))$ is a centered kernel function. Instead of explicitly calculating the feature map of the point $\boldsymbol{x}^\star$, the kernel or similarity to the training-points is calculated. Using above equation to re-generate the kernel matrix K of the training dataset, the denoised similarities $\hat{K}$ are calculated:

$$\hat{K} = \frac{1}{\eta} K H^{\mathrm{T}} H, \tag{5.1.7}$$

where $H \in \mathbb{R}^{s \times N}$ is a subset of the trained hidden units with $s \leq N$. A similar pattern occurs as in equation (5.1.5).

We propose to use these similarities in a kernel smoother approach [78], however other mechanisms are possible. The estimated value $\hat{\boldsymbol{x}}$ for $\boldsymbol{x}^\star$ is now equal to:

$$\hat{\boldsymbol{x}} = \frac{\sum_{j=1}^{S} \tilde{K}(\boldsymbol{x}_j, \boldsymbol{x}^\star)\boldsymbol{x}_j}{\sum_{j=1}^{S} \tilde{K}(\boldsymbol{x}_j, \boldsymbol{x}^\star)}, \tag{5.1.8}$$

where $\tilde{K}(\boldsymbol{x}_j, \boldsymbol{x}^\star)$ is the scaled similarity between 0 and 1 calculated in equation (5.1.6) and a design parameter $S \leq N$, the S closest points based on the similarity $\tilde{K}(\boldsymbol{x}_j, \boldsymbol{x}^\star)$. Kernel smoothing often works with a localized kernel like the RBF kernel, where the second design parameter is the bandwidth $\tilde{\sigma}$.

5.1.3 Experiments

Our goal is to use generative kernel PCA to explore the latent space. Therefore a tool[1] is developed where generative kernel PCA can easily be utilized for new datasets. First, kernel PCA is performed to find the hidden features of the dataset. After choosing an initial hidden unit as a starting point, the values are varied for each component of the hidden unit to explore the latent space. The corresponding newly generated data point in the input space is estimated using the kernel smoother approach.

In the tool, a partial visualization of the latent space projected onto two principal components is shown. We continuously vary the values of the components of the selected hidden unit. This allows the exploration of the extracted latent space by visualizing the resulting variation in the input space. The ability to perform incremental variations aids interpretation of the meaning encoded in the latent space along a chosen direction. In Figure 5.1.1, the interface of our tool is shown .

MNIST Handwritten Digits

As an example, the latent space of the MNIST handwritten digits dataset [131] is explored, where 1000 data points each of digits zero and one are sampled. A Gaussian kernel with bandwidth $\sigma^2 = 50$, $S = 15$ and number of components

[1] Matlab code for the latent space exploration tool is available at `https://www.esat.kuleuven.be/stadius/E/software.php`.

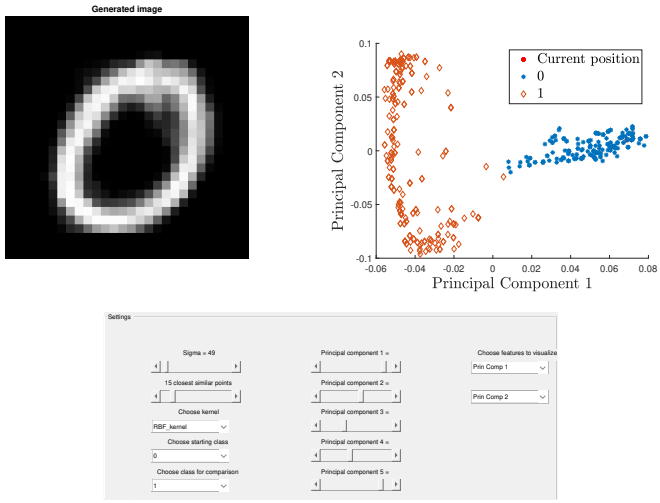
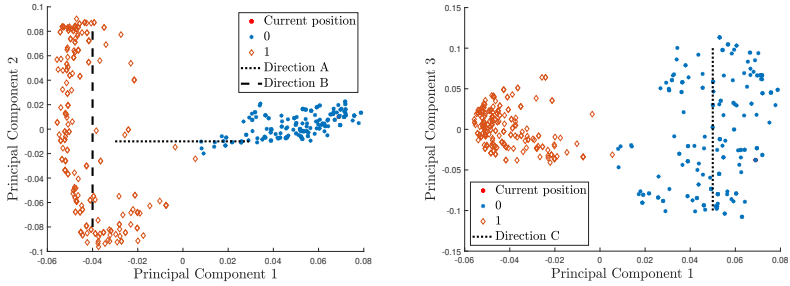


Figure 5.1.1: Interface of the Matlab tool for exploring the latent space. At the bottom, the parameter values and position in the latent space can be chosen. In the top right, the latent space along two selected principal components is shown, and on the left, the newly generated data point in the input space is visualized.

$d = 10$. In Figure 5.1.2, the latent space is shown along the first two principal components as well as the first and third components.

In Figure 5.1.3, digits are generated along the directions indicated on the plots of the latent space in Figure 5.1.2. This allows us to interpret the different regions and the meaning of the principal components. Along direction A, corresponding to the first principal component, we find an interpolation between the regions with digits of zero and one. Direction B seems to correlate with the orientation of the digit. This explains the smaller variation along the second principal component for the zeros as rotating the digit zero has a smaller effect compared to the rotation of digit one. The third direction, corresponding to component 3, seems to be related to squeezing the zeros together, which explains the larger variance for the zeros compared to the ones.

Yale Face Database



(a) (b)

Figure 5.1.2: Latent space of the MNIST digits dataset for the digits 0 and 1. The dotted lines indicate the direction along which new data points are generated. (a) Data projected on the first two principal components (b) Data projected on the first and third principal components.

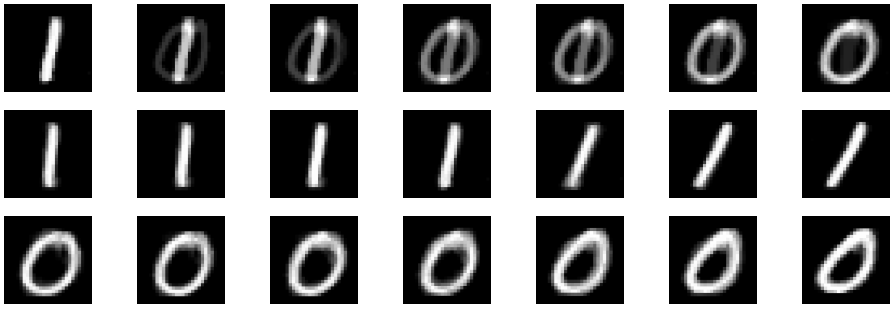
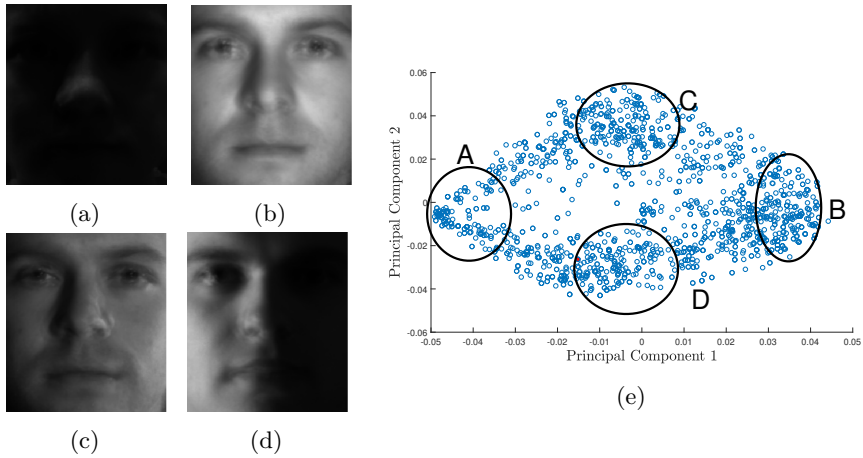


Figure 5.1.3: Exploration of the latent space of the MNIST digits data set. In the top two rows, images are generated along the directions A and B in Figure5.1.2a and the bottom row the images are generated along direction C in Figure 5.1.2b.

Another example of latent space exploration is done on the Extended Yale Face Database B [82], where 1720 data points are sampled. A Gaussian kernel with bandwidth $\sigma^2 = 650$, $S = 45$ and number of components $d = 20$.

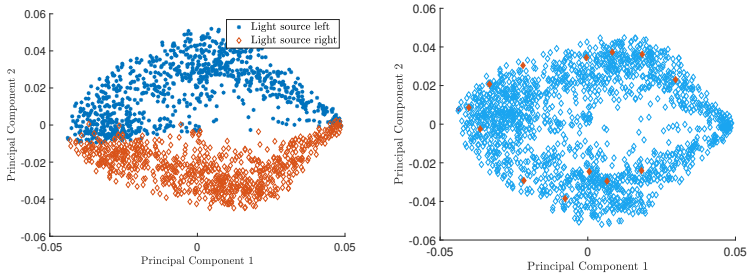
The latent space along the first two principal components is shown in Figure 5.1.4e. Four different regions within the feature space are highlighted from which corresponding images are generated. The dissimilarity between the images in the various regions suggests the components capture different lighting conditions on the subjects.

The tool allows us to gradually move between these different regions and see the



(a) (b) (c) (d) (e)

Figure 5.1.4: Exploring different regions of the latent space of the Yale Face Database. (e) Data projected on the first two principal components for the Yale Face Database. (a)-(d) Generated faces from the different regions (fake images).



(a) (b)

Figure 5.1.5: Latent space of the Yale Facebase database B. (a) Points in orange indicate data points with a negative azimuthal angle between the camera direction and source of illumination, which corresponds to a light source to the right of the subject and vice versa for positive azimuthal angle. (b) Points in red indicate the hidden units of the same subject with lighting from different azimuthal angles.

changes in the input space, as shown in Figure 5.1.6. Moving between regions A and B shows increasing illumination of the subject. We can thus interpret the first principal component as determining the global level of illumination. Note that besides data points without a light source, no variation of the intensity of



Figure 5.1.6: Exploring the space between the regions of the latent space in Figure 5.1.4e. The top row shows images generated between regions A and B, while the bottom row explores the space between regions C and D (fake images).

the lighting was varied while collecting the data for the Yale Face Database B. Only the position of the light source was changed. Generative Kernel PCA thus allows us to control the level of illumination regardless of the position of the light source. The bottom row seems to indicate that the second principal component can be interpreted as the position of the light source. In region C of the feature space, the points are illuminated from the right and region D from the left. This interpretation of the second principal component seems indeed valid from Figure 5.1.5a where the latent space is visualized with labels indicating the position of illumination obtained from the Yale Face Database. Faces with a positive azimuthal angle between the camera direction and the source of illumination are contained in the top half of the figure. This corresponds to a light source left of the subject and vice versa for a negative azimuthal angle. The first and second components are thus disentangled as the level of illumination does not determine whether the light comes from the left or the right. Furthermore, in Figure 5.1.5b the hidden units corresponding to the same subject under different lighting conditions are shown. The elevation of the light source is kept constant at zero elevation, while the azimuthal angle is varied. We see from the plot that the points do not move strictly along the second principal component, but follow a more circular path. This indicates that varying the azimuthal angle correlates with both the first and second principal component, i.e. moving the light source more to the side also decreases the global illumination level as less light can illuminate the face. We conclude that while in the original data set the position of the light source and the level of illumination are correlated, kernel PCA allows us to disentangle these factors and vary them separately when generating new images.

As a further example of generative kernel PCA, interpolation between 2 faces is demonstrated. Kernel PCA is performed on a subset of the database consisting out of 130 facial images of two subjects, the hyperparameters are the same as above. Variation along the fourth principal component results in a smooth interpolation between the two subjects, shown in Figure 5.1.7. We also include an example in the bottom row where the interpolation does not result in a smooth change between the subjects. This illustrates a major limitation of our method as generative kernel PCA predominantly detects global features such as lighting and has difficulty with smaller, local features such as eyes. This stems from the fact that generative kernel PCA relies on the input data to be highly correlated which in this example translates itself to the need for the faces to be aligned with each other.



Figure 5.1.7: Three examples of interpolation between two subjects of the Yale Face Database B along the fourth component (fake images). The uttermost left and right pictures in the rows represent the original faces (real images).

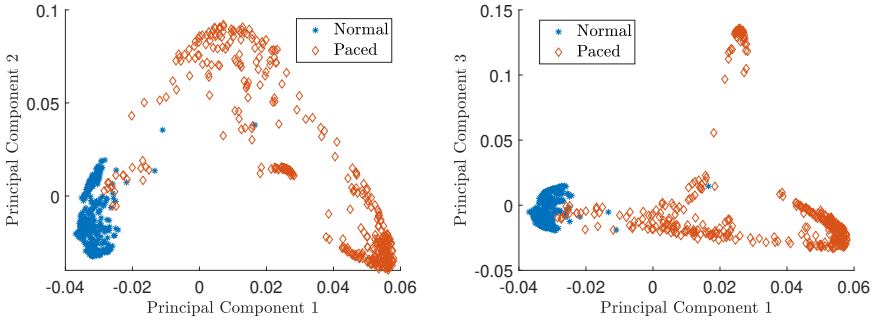
MIT-BIH Arrhythmia database

Besides the previous examples of latent space exploration for image datasets, kernel PCA is also applicable to other types of data. In this section, the MIT-BIH Arrhythmia dataset [163] is consisting out of ECG signals. The goal is to demonstrate the use of kernel PCA to extract interpretable directions in the

latent feature space of the ECG signals. This would allow a clinical expert to gain insight and trust in the features extracted by the model. Similar research was previously done by [236] where they investigated the use of disentangled variational autoencoders to extract interpretable ECG embeddings. A similar approach is used to preprocess the data as in [236].

The signals from the patients with identifiers 101, 106, 103, and 105 are used for the normal beat signals and the data of patients 102, 104, 107, 217 for the paced beat signals. This results in a total of 785 beat patterns which are processed through a peak detection program [212]. The ECG signal is first passed through a fifth-order Butterworth bandpass filter with a lower cutoff frequency of 1Hz and an upper cutoff frequency of 60Hz. The ECG beats are sampled at 360Hz and a window of 0.5 seconds is taken around each R-wave resulting in 180 samples per epoch. A regular Gaussian kernel with bandwidth $\sigma^2 = 10$ is used, with $S = 10$. The first 10 principal components are used in the reconstruction.

In Figure 5.1.8 the latent feature space projected on the first two principal components is shown. Kernel PCA is also able to separate between the normal and paced beats.



(a) (b)

Figure 5.1.8: The latent space for 785 ECG beat signals of the MIT-BIH Arrhythmia dataset projected on different principal components. The hidden units of both normal and paced heartbeats are shown.

Figure 5.1.9 shows the result in the input space of moving along the first principal components in the latent feature space. As original base point we take a normal beat signal, i.e. corresponding to a hidden unit on the bottom right of Figure 5.1.8a. The smooth transition between the beat patterns allows for the interpretation of the first principal components. This allows a clinical expert

to understand on what basis the paced beats are separated by the principal components and if this basis has a physiological meaning. To investigate the separated region of the latent space at the top of Figure 5.1.8b we start from a paced beat pattern and vary along the third principal component. This allows us to see which sort of heartbeat patterns are responsible for this specific distribution in the latent space.

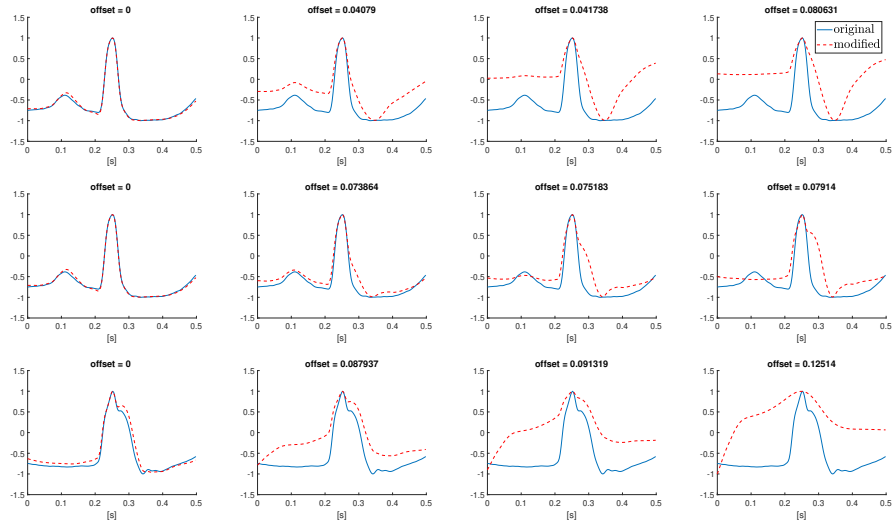


Figure 5.1.9: Exploring the first three principal components of the latent feature space for the MIT-BIH arrhythmia database for normal and paced beats. The red line represents the newly generated data point compared to the original point depicted in blue. The top, middle, and bottom rows represent the variation along the first, second and third components respectively. The top and middle rows start with a normal heartbeat pattern and the bottom row with a paced signal.

5.1.4 Novelty detection

As a final illustration of latent space exploration using generative kernel PCA, we consider an application within the context of novelty detection. We use the reconstruction error in feature space as a measure of novelty [102], where Hoffmann shows the metric demonstrates a competitive performance on synthetic distributions and real-world data sets. The novelty score is calculated for all data points, where 20% of data points with the largest novelty score are considered novel. These points typically reside in low-density regions of the latent space

and are highlighted as interesting regions to explore using the tool. we consider 1000 instances of the digit zero from the MNIST dataset. After performing kernel PCA with the same parameters as in the previous section, we explore the latent space around the detected novel patterns. The data projected on the first two principal components is shown in Figure 5.1.10.

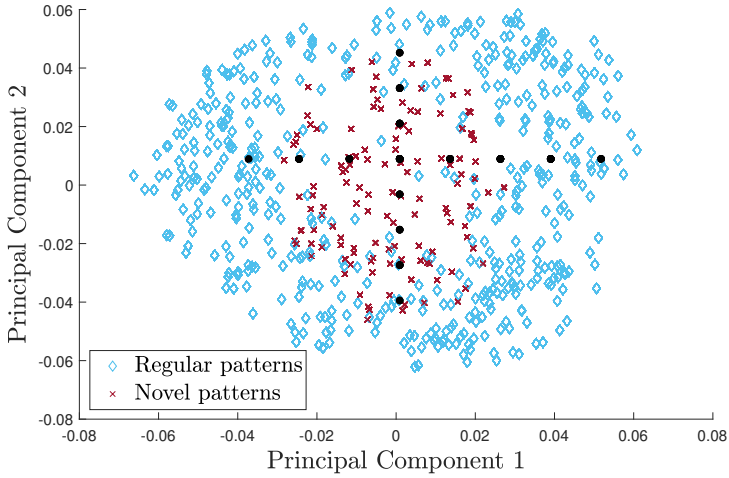


Figure 5.1.10: The latent space for 1000 zeros of the MNIST digits data set. The central cluster of points consists out of data points with a high novelty score, this corresponds to a low-density region in the latent space. The black dots indicate the points in latent space which are sampled.

The generated images from the positions indicated by the black dots in Figure 5.1.10 are shown in Figure 5.1.11. The first row allows us to interpret the first principal component as moving from a thin round zero towards a more closed digit. The middle of the latent space is where the novel patterns are located which seems to indicate most zeros are either thin and wide or thick and narrow. A low amount of zeros in the data set are thick and wide or very thin and narrow. The bottom row of Figure 5.1.11 gives the interpretation for the second principal component as rotating the digit. The novel patterns seem to be clustered more together and as such have a less obvious orientation. Important to note is that we only look at the first 2 components for the interpretation, while in practice the novelty detection method considers all 20 components.

The above experiment shows that latent space exploration methods can give additional insights for novelty detection. Both the generating mechanism, as well as the novelty detection make use of the kernel PCA formulation. The

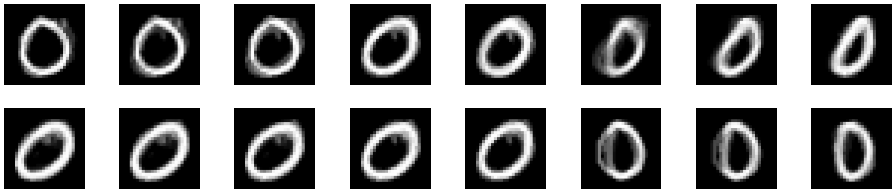


Figure 5.1.11: Exploration of the latent space in Figure 5.1.10. The top row indicates the points generated from the horizontal black dots, while the bottom row corresponds to the vertical positions.

two methods naturally complement each other: novelty detection provides interesting regions in the latent space to explore, at the same time helps the generative mechanism in interpreting why certain points are considered novel.

5.2 Generative restricted kernel machines: a framework for multi-view generation and disentangled feature learning

5.2.1 Introduction

In this section, we continue with the generative RKM formulation. When working with more complex data like images, it is known that convolutional type feature maps outperform classical kernel functions. In this section, the use of explicit feature maps, in the form of neural networks, is introduced. We propose a novel training procedure that jointly learns the features and a shared subspace representation. Moreover, we introduce multi-view generation into the framework. Many datasets are composed of different representations of the data, also called views. Views can correspond to different modalities such as sounds, images, videos, sequences of previous frames, etc. Although each view could individually be used for learning tasks, exploiting information from all views together could improve the learning quality [183, 143, 41].

We have the following contributions: 1) A novel joint multi-view generative model based on the RKM framework where multiple views of the data can be generated simultaneously. 2) Two methods are discussed for computing the pre-image of the feature vectors: with the feature map explicitly known or unknown. We show that the mechanism is flexible to incorporate both kernel-based and (deep convolutional) neural network-based models within the same setting. 3)

When using explicit feature maps, we propose a training algorithm that jointly performs the feature selection and learns the common-subspace representation in the same procedure. 4) Qualitative and quantitative experiments demonstrate that the model is capable of generating good quality images of natural objects. Further illustrations on multi-view datasets exhibit the potential of the model. Thanks to the orthogonality of eigenvectors of the kernel matrix, the learned latent variables are uncorrelated. This resembles a disentangled representation, which makes it possible to generate data with specific characteristics.

5.2.2 Related work

Latent space models were studied in several other works, where multiple links with disentanglement are made. VAEs [122] have become a popular framework among different generative models as they provide more theoretically well-founded and stable training than GANs [88]. Learning a VAE amounts to the optimization of an objective balancing the quality of samples that are autoencoded through a stochastic encoder-decoder pair, measured by the reconstruction error, while encouraging the latent space to follow a fixed prior distribution, often the Gaussian distribution. In β-VAEs [99], an adjustable hyperparameter β is introduced that balances quality of samples and latent space constraints with reconstruction accuracy. The choice of parameter $\beta = 1$ corresponds to the original VAE formulation. Further, they show that with $\beta > 1$ (more emphasis on the latent variables to be Gaussian distributed) the model is capable of learning a more disentangled latent representation of the data. In [31], the effect of the β term is analyzed more in-depth. It was suggested that the stronger pressure for the posterior to match the factorized unit Gaussian prior puts extra constraints on the implicit capacity of the latent bottleneck [99]. Chen et al. [42] show a decomposition of the variational lower bound that can be used to explain the success of the β-VAE [99] in learning disentangled representations. The authors claim that the total correlation, which forces the model to find statistically independent factors in the data distribution, is the most important term in this decomposition. The role of disentanglement was also studied in GANs, where the InfoGAN [43] is one of the most known works.

The most common approach of joint multimodal/multiview learning with deep neural networks is to share the top of hidden layers in modality-specific networks. Srivastava and Salakhutdinov [218] proposed a Deep Boltzmann Machine for learning multimodal data. The multimodal DBM learns a joint density model over the space of multimodal inputs by sharing the hidden units of the last layer. Examples of joint multimodal training for VAEs are [226, 249]. The work of [226] introduced the joint multi-modal VAE, which learns the common

distribution using a joint inference network. The authors use an ELBO objective with two additional divergence terms to minimize the distance between the uni-modal and the multi-modal importance distributions. The MVAE of [249] uses a product of experts formulation and sub-sampled training paradigm to solve the multi-modal inference problem.

In contrast to classical VAE architectures, the proposed model introduces an orthogonal interconnection matrix U motivated by the RKM formulation. The model thus finds an 'optimal' linear subspace of the latent space given by the eigendecomposition. In this section, we argue that this orthogonality leads to better disentanglement and generation quality.

5.2.3 Generative restricted kernel machines framework

The proposed Gen-RKM framework consists of two phases: a training phase and a generation phase which occurs one after the other.

Training phase of the RKM

Similar to Energy-Based Models (EBMs, see [133] for details), the RKM objective function captures dependencies between variables by associating a scalar energy to each configuration of the variables. Learning consists of finding an energy function in which the observed configurations of the variables are given lower energies than unobserved ones. Note that the schematic representation of Gen-RKM model, as shown in Figure 5.2.1 is similar to Discriminative RBMs [128] and the objective function $\mathcal{J}_t$ (defined below) has an energy form similar to RBMs with additional regularization terms. The latent space dimension in the RKM setting has a similar interpretation as the number of hidden units in a Restricted Boltzmann Machine, where in the specific case of the RKM these hidden units are uncorrelated.

We assume a dataset $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ with $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y}_i \in \mathbb{R}^p$ consisting of N data points. Here $\boldsymbol{y}_i$ may represent an additional view of $\boldsymbol{x}_i$, e.g., an additional image from a different angle, the caption of an image or a class label. We start with the RKM interpretation of Kernel PCA, which gives an upper bound on the equality constrained Least-Squares Kernel PCA objective function [224]. Applying the feature-maps $\phi_1 : \Omega_{\boldsymbol{x}} \mapsto \mathcal{H}_{\boldsymbol{x}}$ and $\phi_2 : \Omega_{\boldsymbol{y}} \mapsto \mathcal{H}_{\boldsymbol{y}}$ to the input data points, where $\mathcal{H}_{\boldsymbol{x}}, \mathcal{H}_{\boldsymbol{y}}$ are the corresponding Reproducing Kernel Hilbert Spaces (RKHS) of the feature-maps respectively; the training objective function $\mathcal{J}_t$ for

generative RKM is given by[2]:

$$\begin{aligned}\mathcal{J}_t = \sum_{i=1}^{N} \left\{ -\phi_1(\boldsymbol{x}_i)^\top U \boldsymbol{h}_i - \phi_2(\boldsymbol{y}_i)^\top V \boldsymbol{h}_i + \frac{1}{2} \boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i \right\} \\ + \frac{\eta_1}{2} \operatorname{Tr}(U^\top U) + \frac{\eta_2}{2} \operatorname{Tr}(V^\top V),\end{aligned} \tag{5.2.1}$$

where $U \in \mathbb{R}^{d_f \times s}$, $V \in \mathbb{R}^{p_f \times s}$ are the unknown interconnection matrices, $\Lambda \succ 0$ the unknown diagonal matrix and $\boldsymbol{h}_i \in \mathbb{R}^s$ are the latent variables modeling a common subspace $\mathcal{H} \subseteq \mathcal{H}_{\boldsymbol{x}} \bigoplus \mathcal{H}_{\boldsymbol{y}}$ between the two feature spaces. (see Figure 5.2.1). Given $\eta_1 > 0$ and $\eta_2 > 0$ as regularization parameters, the stationary points of $\mathcal{J}_t$ are given by:

$$\begin{cases} \frac{\partial \mathcal{J}_t}{\partial \boldsymbol{h}_i} = 0 \implies \Lambda \boldsymbol{h}_i = U^\top \phi_1(\boldsymbol{x}_i) + V^\top \phi_2(\boldsymbol{y}_i), \ \forall i \\ \frac{\partial \mathcal{J}_t}{\partial U} = 0 \implies U = \frac{1}{\eta_1} \sum_{i=1}^{N} \phi_1(\boldsymbol{x}_i) \boldsymbol{h}_i^\top, \\ \frac{\partial \mathcal{J}_t}{\partial V} = 0 \implies V = \frac{1}{\eta_2} \sum_{i=1}^{N} \phi_2(\boldsymbol{y}_i) \boldsymbol{h}_i^\top. \end{cases} \tag{5.2.2}$$

Substituting U and V in the first equation above, denoting the diagonal matrix $\Lambda = \operatorname{diag}\{\lambda_1, \ldots, \lambda_s\} \in \mathbb{R}^{s \times s}$ with $s \leq N$, yields the following eigenvalue problem:

$$\left[\frac{1}{\eta_1} K_1 + \frac{1}{\eta_2} K_2 \right] \boldsymbol{h}^\top = \boldsymbol{h}^\top \Lambda, \tag{5.2.3}$$

where $h = \begin{bmatrix} \boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \end{bmatrix} \in \mathbb{R}^{s \times N}$ with $s \leq N$ is the number of selected principal components and $K_1, K_2 \in \mathbb{R}^{N \times N}$ are the kernel matrices corresponding to data sources. For simplicity, we assume that have only two data sources (namely $\Omega_{\boldsymbol{x}}$ and $\Omega_{\boldsymbol{y}}$) available for learning. However, the procedure could be extended to multiple data-sources. For the M views or data-sources, this yields the training problem: $\left[\sum_{\ell=1}^{M} \frac{1}{\eta_\ell} K_\ell \right] \boldsymbol{h}^\top = \boldsymbol{h}^\top \Lambda$. Based on Mercer's theorem [157], positive-definite kernel functions $k_1 : \Omega_{\boldsymbol{x}} \times \Omega_{\boldsymbol{x}} \mapsto \mathbb{R}$, $k_2 : \Omega_{\boldsymbol{y}} \times \Omega_{\boldsymbol{y}} \mapsto \mathbb{R}$ can be defined such that $k_1(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi_1(\boldsymbol{x}_i), \phi_1(\boldsymbol{x}_j) \rangle_{\mathcal{H}_{\boldsymbol{x}}}$, and $k_2(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \phi_2(\boldsymbol{y}_i), \phi_2(\boldsymbol{y}_j) \rangle_{\mathcal{H}_{\boldsymbol{y}}}$, $\forall i, j = 1, \ldots, N$ forms the elements of corresponding kernel matrices. The feature maps ϕ_1 and ϕ_2, mapping the input data to the high-dimensional feature space (possibly infinite) are implicitly defined by kernel functions. Typical examples of such kernels are given by the Gaussian RBF kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 / (2\sigma^2)}$ or the Laplace kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 / \sigma}$ just to name a few [209]. However, one can also define explicit feature maps, still preserving the positive-definiteness of the

[2]For convenience, it is assumed that the feature vectors are centered in the feature space $\Omega_{\boldsymbol{x}}, \Omega_{\boldsymbol{y}}$ using $\tilde{\phi}(\boldsymbol{x}) := \phi(\boldsymbol{x}) - \frac{1}{N} \sum_{i=1}^{N} \phi(\boldsymbol{x}_i)$.

kernel function by construction [225]. Equation 5.2.3 corresponds to a kernel PCA operation. In this spirit, we thus find an orthogonal interconnection matrix U that is the optimal linear subspace of the latent space given by the eigendecomposition.

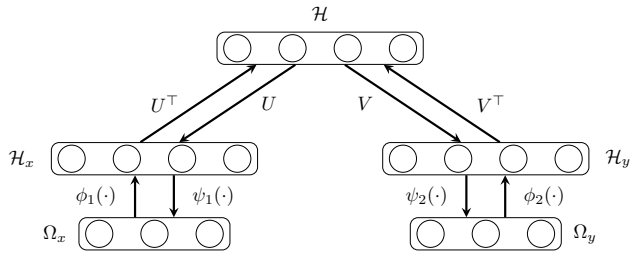


Figure 5.2.1: Gen-RKM schematic representation modeling a common subspace $\mathcal{H} \subseteq \mathcal{H}_{\boldsymbol{x}} \bigoplus \mathcal{H}_{\boldsymbol{y}}$ between two data sources $\Omega_{\boldsymbol{x}}$ and $\Omega_{\boldsymbol{y}}$. The ϕ_1, ϕ_2 are the feature maps ($\mathcal{H}_{\boldsymbol{x}}$ and $\mathcal{H}_{\boldsymbol{y}}$ represents the RKHS) corresponding to the two data sources. While ψ_1, ψ_2 represents the pre-image maps. The interconnection matrices U, V capture the dependencies between latent variables and the mapped data sources.

Generation

In this section, we derive the equations for the generative mechanism. RKMs resembling energy-based models, the inference consists in clamping the value of observed variables and finding configurations of the remaining variables that minimizes the energy [133]. Given the learned interconnection matrices U and V, and a given latent variable $\boldsymbol{h}^\star$, consider the following generation objective function $\mathcal{J}_g$:

$$\mathcal{J}_g = -\hat{\phi}_1(\boldsymbol{x}^\star)^\top U \boldsymbol{h}^\star - \hat{\phi}_2(\boldsymbol{y}^\star)^\top V h^* + \frac{1}{2}\hat{\phi}_1(\boldsymbol{x}^\star)^\top \hat{\phi}_1(\boldsymbol{x}^\star) + \frac{1}{2}\hat{\phi}_2(\boldsymbol{y}^\star)^\top \hat{\phi}_2(\boldsymbol{y}^\star),$$

with an additional regularization term on data sources. With slight abuse of notation, we denote the generated feature vectors by $\hat{\phi}_1(\boldsymbol{x}^\star)$ and $\hat{\phi}_2(\boldsymbol{y}^\star)$ given the corresponding latent variable $\boldsymbol{h}^\star$, to distinguish from the feature vectors corresponding to training data points (see (5.2.1)). The given latent variable $\boldsymbol{h}^\star$ can be the corresponding latent code of a training point, a newly sampled hidden unit or a specifically determined one. Above cases correspond to generating the reconstructed visible unit, generating a random new visible unit or exploring the latent space by carefully selecting hidden units respectively. The stationary

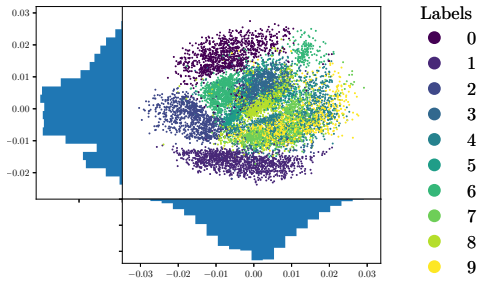


Figure 5.2.2: MNIST: Scatter plot of latent variable distribution when trained on 10000 images ($s = 2$). Training was unsupervised (i.e. one-view) and labels are only used to color the plot. The latent space resembles a Gaussian distribution centered around 0, where the various digits are clustered together.

points of $\mathcal{J}_g$ are characterized by:

$$\begin{cases} \frac{\partial \mathcal{J}_g}{\partial \hat{\phi}_1(\boldsymbol{x}^\star)} = 0 \implies \hat{\phi}_1(\boldsymbol{x}^\star) = U\boldsymbol{h}^\star, \\ \frac{\partial \mathcal{J}_g}{\partial \hat{\phi}_2(\boldsymbol{y}^\star)} = 0 \implies \hat{\phi}_2(\boldsymbol{y}^\star) = V\boldsymbol{h}^\star. \end{cases} \tag{5.2.4}$$

Using U and V from (5.2.2), we obtain the generated feature vectors:

$$\begin{aligned} \hat{\phi}_1(\boldsymbol{x}^\star) &= \left(\frac{1}{\eta_1} \sum_{i=1}^{N} \phi_1(\boldsymbol{x}_i) \boldsymbol{h}_i^\top \right) \boldsymbol{h}^\star, \\ \hat{\phi}_2(\boldsymbol{y}^\star) &= \left(\frac{1}{\eta_2} \sum_{i=1}^{N} \phi_2(\boldsymbol{y}_i) \boldsymbol{h}_i^\top \right) \boldsymbol{h}^\star. \end{aligned} \tag{5.2.5}$$

To obtain the generated data, now one should compute the inverse images of the feature maps $\hat{\phi}_1(\cdot)$ and $\hat{\phi}_2(\cdot)$ in the respective input spaces, i.e., solve the *pre-image problem*. We seek to find the functions $\psi_1 \colon \mathcal{H} \mapsto \Omega_{\boldsymbol{x}}$ and $\psi_2 \colon \mathcal{H} \mapsto \Omega_{\boldsymbol{y}}$ corresponding to the two data-sources, such that $(\psi_1 \circ \hat{\phi}_1)(\boldsymbol{x}^\star) \approx \boldsymbol{x}^\star$ and $(\psi_2 \circ \hat{\phi}_2)(\boldsymbol{y}^\star) \approx \boldsymbol{y}^\star$, where $\hat{\phi}_1(\boldsymbol{x}^\star)$ and $\hat{\phi}_2(\boldsymbol{y}^\star)$ are given using (5.2.5).

When using kernel methods, explicit feature maps are not necessarily known. Commonly used kernels such as the radial-basis function and polynomial kernels map the input data to a very high dimensional feature space. Hence finding the pre-image, in general, is known to be an ill-conditioned problem [160]. However, various approximation techniques have been proposed [29, 127, 104, 243] which could be used to obtain the approximate pre-image $\hat{\boldsymbol{x}}$ of $\hat{\phi}_1(\boldsymbol{x}^\star)$. In section 5.2.4,

we employ one such technique to demonstrate the applicability in our model, and consequently generate the multi-view data. One could also define explicit pre-image maps. In section 5.2.4, we define parametric pre-image maps and learn the parameters by minimizing the appropriately defined objective function. The next section describes the above two pre-image methods for both cases, i.e., when the feature map is explicitly known or unknown, in greater detail.

5.2.4 The proposed algorithm with implicit & explicit feature maps

Implicit feature map

As noted in the previous section, since $\boldsymbol{x}^\star$ may not exist, we find an approximation $\hat{\boldsymbol{x}}$. A possible technique was shown in the previous section. Left multiplying (5.2.5) by $\hat{\phi}_1(\boldsymbol{x}_i)^\top$ and $\hat{\phi}_2(\boldsymbol{y}_i)^\top$, $\forall i = 1, \ldots, N$, we obtain:

$$k_{\boldsymbol{x}^\star} = \frac{1}{\eta_1} K_1 \boldsymbol{h}^\top \boldsymbol{h}^\star, \quad K_{\boldsymbol{y}^\star} = \frac{1}{\eta_2} K_2 \boldsymbol{h}^\top \boldsymbol{h}^\star, \tag{5.2.6}$$

where, $k_{\boldsymbol{x}^\star} = [k(\boldsymbol{x}_1, \boldsymbol{x}^\star), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x}^\star)]^\top$ represents the *similarities* between $\hat{\phi}_1(\boldsymbol{x}^\star)$ and training data points in the feature space, and $K_1 \in \mathbb{R}^{N \times N}$ represents the centered kernel matrix of $\Omega_{\boldsymbol{x}}$. Similar conventions follow for $\Omega_{\boldsymbol{y}}$ respectively. Using the *kernel-smoother* method [95], the pre-images are given by:

$$\begin{aligned} \hat{\boldsymbol{x}} &= (\psi_1 \circ \hat{\phi}_1)(\boldsymbol{x}^\star) = \frac{\sum_{j=1}^{n_r} \tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star) \boldsymbol{x}_j}{\sum_{j=1}^{n_r} \tilde{k}_1(\boldsymbol{x}_j, \boldsymbol{x}^\star)}, \\ \hat{\boldsymbol{y}} &= (\psi_2 \circ \hat{\phi}_2)(\boldsymbol{y}^\star) = \frac{\sum_{j=1}^{n_r} \tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star) \boldsymbol{y}_j}{\sum_{j=1}^{n_r} \tilde{k}_2(\boldsymbol{y}_j, \boldsymbol{y}^\star)}, \end{aligned} \tag{5.2.7}$$

where $\tilde{k}_1(\boldsymbol{x}_i, \boldsymbol{x}^\star)$ and $\tilde{k}_2(\boldsymbol{y}_i, \boldsymbol{y}^\star)$ are the scaled similarities (see (5.2.7)) between 0 and 1 and n_r the number of closest points based on the similarity defined by kernels $\tilde{k}_1$ and $\tilde{k}_2$.

Explicit feature map

While using an explicit feature map, Mercer's theorem is still applicable due to the positive semi-definiteness of the kernel function by construction, thereby allowing the derivation of (5.2.3). In the experiments, we use a set of (convolutional) neural networks as the parametric feature maps $\phi_{\boldsymbol{\theta}}(\cdot)$.

Algorithm 5.2.1 Gen-RKM

Input: $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, η_1, η_2, feature map $\phi_j(\cdot)$ - explicit *or* implicit via kernels $k_j(\cdot,\cdot)$, for $j \in \{1,2\}$
Output: Generated data $\boldsymbol{x}^\star$, $\boldsymbol{y}^\star$

procedure TRAIN
  if $\phi_j(\cdot)$ = Implicit **then**
    Solve the eigendecomposition in (5.2.3)
    Select the s first principal components
  else if $\phi_j(\cdot)$ = Explicit **then**
    while not converged **do**
      $\{\boldsymbol{x}, \boldsymbol{y}\} \leftarrow$ {Get mini-batch}
      $\phi_1(\boldsymbol{x}) \leftarrow \boldsymbol{x}$; $\phi_2(\boldsymbol{y}) \leftarrow \boldsymbol{y}$ ▷ Get embeddings
      do steps 3-4
      $\{\hat{\phi}_1(\boldsymbol{x}), \hat{\phi}_2(\boldsymbol{y})\} \leftarrow \boldsymbol{h}$ ((5.2.5))
      $\{\boldsymbol{x}, \boldsymbol{y}\} \leftarrow \{\psi_1(\hat{\phi}_1(\boldsymbol{x})), \psi_2(\hat{\phi}_2(\boldsymbol{y}))\}$ ▷ Pre-image map
      $\Delta\{\boldsymbol{\theta}, \boldsymbol{\zeta}\} \propto -\nabla_{\{\boldsymbol{\theta},\boldsymbol{\zeta}\}} \mathcal{J}_c$ ▷ Update parameters
    end while
  end if
end procedure

procedure GENERATION
  Select $\boldsymbol{h}^\star$
  if $\phi_j(\cdot)$ = Implicit **then**
    Set hyperparameter: n_r
    Compute $K_{\boldsymbol{x}^*}$, $K_{\boldsymbol{y}^*}$ ((5.2.6))
    Get $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ ((5.2.7))
  else if $\phi_j(\cdot)$ = Explicit **then**
    do steps 10-11
  end if
end procedure

Another (transposed convolutional) neural network is used for the pre-image map $\psi_{\boldsymbol{\zeta}}(\cdot)$ [69]. The network parameters $\{\boldsymbol{\theta}, \boldsymbol{\zeta}\}$ are learned by minimizing the reconstruction errors $\mathcal{L}_1(x, \psi_{1_{\boldsymbol{\zeta}_1}}(\hat{\phi}_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}))) = \frac{1}{N}\sum_{i=1}^N \|\boldsymbol{x}_i - \psi_{1_{\boldsymbol{\zeta}_1}}(\hat{\phi}_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i))\|_2^2$ for the first view and $\mathcal{L}_2(y, \psi_{2_{\boldsymbol{\zeta}_2}}(\hat{\phi}_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}))) = \frac{1}{N}\sum_{i=1}^N \|\boldsymbol{y}_i - \psi_{2_{\boldsymbol{\zeta}_2}}(\hat{\phi}_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i))\|_2^2$ for the second view, however, in principle, one can use any other loss appropriate to the dataset. Here $\hat{\phi}_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i)$ and $\hat{\phi}_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i)$ are given by (5.2.5), i.e., the generated points in feature space from the subspace $\mathcal{H}$. Adding the loss function directly into the objective function $\mathcal{J}_t$ is not suitable for minimization. Instead, we use the stabilized objective function defined as $\mathcal{J}_{stab} = \mathcal{J}_t + \frac{c_{\text{stab}}}{2}\mathcal{J}_t^2$, where $c_{stab} \in \mathbb{R}^+$ is the regularization constant [224]. This tends to push the objective

function $\mathcal{J}_t$ towards zero, which is also the case when substituting the solutions $\lambda_i, \boldsymbol{h}_i$ back into $\mathcal{J}_t$ (see E.2 for details). The combined training objective is given by:

$$\min_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\zeta_1,\zeta_2} \mathcal{J}_c = \mathcal{J}_{stab} + \frac{\gamma}{2N}\Bigg(\sum_{i=1}^{N}\big[\mathcal{L}_1(\boldsymbol{x}_i, \psi_{1_{\zeta_1}}(\hat{\phi}_{1_{\boldsymbol{\theta}_1}}(\boldsymbol{x}_i))) + \mathcal{L}_2(\boldsymbol{y}_i, \psi_{2_{\zeta_2}}(\hat{\phi}_{2_{\boldsymbol{\theta}_2}}(\boldsymbol{y}_i)))\big]\Bigg),$$

where $\gamma \in \mathbb{R}^+$ is a regularization constant to control the stability with reconstruction accuracy. In this way, we combine feature-selection and subspace learning within the same training procedure.

In the objective of the VAE [122], an extra term in the form of the Kullback-Leibler divergence between the encoder's distribution and a unit Gaussian is added as a prior on the latent variables. This ensures the latent space is smooth and without discontinuities, which is essential for a good generation. By interpreting kernel PCA within the LS-SVM setting [225], the PCA analysis can take the interpretation of a one-class modeling problem with zero target value around which one maximizes the variance [223]. When choosing a good feature map, one expects the latent variables to be normally distributed around zero. As a result, the latent space of the Gen-RKM is continuous, allowing easy random sampling and interpolation (see Figure 5.2.2). Kernel PCA gives uncorrelated components in feature space [23]. While the standard PCA does not give a good disentangled representation for images [72, 99]. By designing a good kernel (through appropriate feature-maps) and doing kernel PCA, it is possible to get a disentangled representation for images as we demonstrate in Figure 5.2.7.

5.2.5 The Gen-RKM algorithm

Based on the previous discussion, we propose a novel procedure, called the Gen-RKM algorithm, combining kernel learning and generative models. We show that this procedure is efficient to train and evaluate. The training procedure simultaneously involves feature selection, common-subspace learning, and pre-image map learning. This is achieved via an optimization procedure where one iteration involves an eigendecomposition of the kernel matrix which is composed of the features from various views (see (5.2.3)). The latent variables are given by the eigenvectors, from which a pre-image map reconstructs the generated sample. Figure 5.2.1 shows a schematic representation of the algorithm when two data sources are available.

Thanks to training in m mini-batches, this procedure is scalable to large datasets (sample size N) with training time scaling super-linearly with $T_m = c\frac{N^\gamma}{m^{\gamma-1}}$, instead of $T_k = cN^\gamma$, where $\gamma \approx 3$ for algorithms based on decomposition methods, with some proportionality constant c. The training time could be further reduced by computing the covariance matrix (size $(d_f + p_f) \times (d_f + p_f)$) instead of a kernel matrix (size $\frac{N}{m} \times \frac{N}{m}$), when the sum of the dimensions of the feature-spaces is less than the samples in mini-batch i.e. $d_f + p_f \leq \frac{N}{m}$. When using neural networks as feature maps, d_f and p_f correspond to the number of neurons in the output layer, which are chosen as hyperparameters by the practitioner. Eigendecomposition of this smaller covariance matrix would yield U and V as eigenvectors, where computing the $\boldsymbol{h}_i$ involves only matrix-multiplication which is readily parallelizable on modern GPUs:

$$\begin{bmatrix} \frac{1}{\eta_1}\phi_{\boldsymbol{x}}\phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_1}\phi_{\boldsymbol{x}}\phi_{\boldsymbol{y}}^\top \\ \frac{1}{\eta_2}\phi_{\boldsymbol{y}}\phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_2}\phi_{\boldsymbol{y}}\phi_{\boldsymbol{y}}^\top \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} \Lambda, \tag{5.2.8}$$

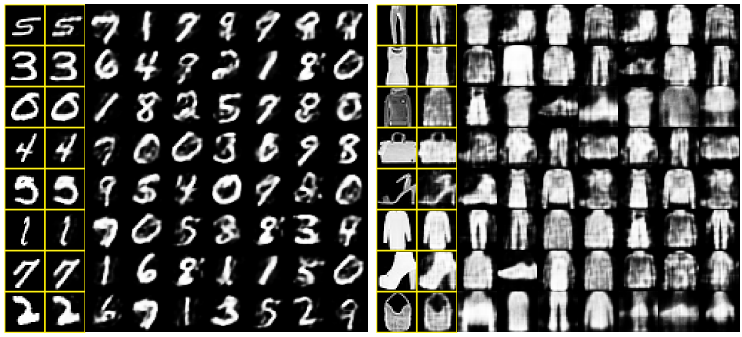
where:

$$\phi_{\boldsymbol{x}} \coloneqq [\phi_1(\boldsymbol{x}_1), \ldots, \phi_1(\boldsymbol{x}_N)]\,,$$

$$\phi_{\boldsymbol{y}} \coloneqq [\phi_2(\boldsymbol{y}_1), \ldots, \phi_2(\boldsymbol{y}_N)]\,.$$

5.2.6 Experiments

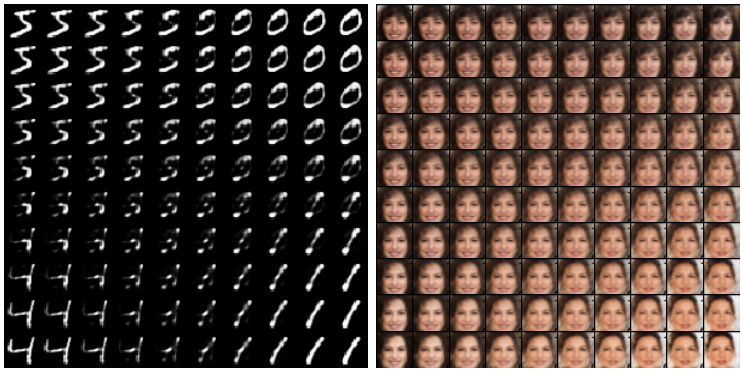
To demonstrate the applicability of the proposed framework and algorithm, we trained the Gen-RKM model on a variety of datasets commonly used to evaluate generative models: MNIST [132], Fashion-MNIST [251], CIFAR-10 [123], CelebA [145], Sketchy [205], Dsprites [154] and Teapot [72]. The proposed method adheres to both a primal and dual formulation to incorporate both kernel-based methods as well as neural networks based models in the same setting. The convolutional neural networks are used as explicit feature maps which are known to outperform kernel based feature maps on the image datasets. Moreover, by using explicit feature maps we demonstrate the capability of the algorithm to jointly learn the feature map and shared subspace representation. For completeness, we also give an illustration when using implicit feature maps through the Gaussian kernel in Figure 5.2.6.



(a) MNIST (b) Fashion-MNIST



(c) CIFAR-10 (d) CelebA



(e) Bilinear interpolation (f) Bilinear interpolation

Figure 5.2.3: Generated samples from the model using CNN as an explicit feature map in the kernel function. In (a), (b), (c), (d) the yellow boxes in the first column show training examples, and the adjacent boxes show the reconstructed samples. The other images (columns 3-6) are generated by random sampling from the fitted distribution over the learned latent variables. (e) and (f) show the generated images through bilinear interpolations in the latent space. The faces in the first column of Figure (d) are real, the other faces shown above are fake.

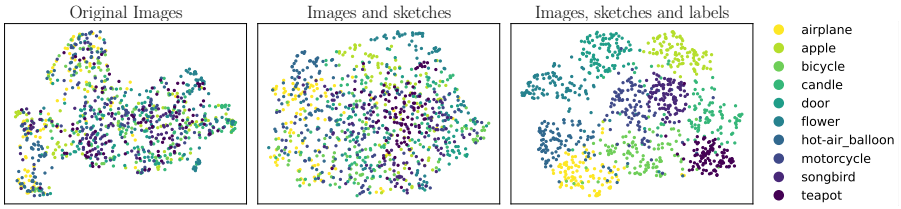


Figure 5.2.4: Learned latent space visualization of the Sketchy dataset in 1, 2 and 3-view Gen-RKM setting by using an UMAP embedding. [156]

In our experiments, we fit a Gaussian mixture model (GMM) with l components to the latent variables of the training set and randomly sample a new point $\boldsymbol{h}^\star$ for generating views. In case of explicit feature maps, we define $\phi_{1_{\boldsymbol{\theta}_1}}$ and $\psi_{1_{\zeta_1}}$ as convolution and transposed-convolution neural networks, respectively [69]; and $\phi_{2_{\boldsymbol{\theta}_2}}$ and $\psi_{1_{\zeta_2}}$ as fully-connected networks. The particular architecture details are outlined in Table E.3.2 in Appendix E.3. The training procedure in the case of explicitly defined maps consists of minimizing $\mathcal{J}_c$ using the Adam optimizer [121] to update the weights and biases. To speed up learning, we subdivided the datasets into m mini-batches, and within each iteration of the optimizer, (5.2.3) is solved to model the subspace $\mathcal{H}$. Information on the datasets and hyperparameters used for the experiments is given in Table E.3.1 in the Appendix. A comparison of the average training time is given in Table E.4.1 in the Appendix.

Random generation. 1) Qualitative examples: Figure 5.2.3 shows the generated images using a convolutional neural network and transposed-convolutional neural network as the feature map and pre-image map respectively. The first column in yellow boxes shows the training samples and the second column on the right shows the reconstructed samples. The other images shown are generated by random sampling from a GMM over the learned latent variables. Notice that the reconstructed samples are of better quality visually than the other images generated by random sampling. To demonstrate that the model has not merely memorized the training examples, we show the generated images via bilinear-interpolations in the latent space in Figure 5.2.3e and Figure 5.2.3f.

2) Quantitative comparison: We compare the proposed model with the standard VAE [122], β-VAE [122], β-TCVAE [42] and Info-GAN [43]. For the Info-GAN, batch normalization is added for training stability. As suggested by the authors, we keep $\alpha = \gamma = 1$ and only modify the hyperparameter β for the β-TCVAE model. Determination of the β hyperparameter is done by starting from values in the range of the parameters suggested in the authors' reference

Table 5.2.1: FID Scores [98] for randomly generated samples (smaller is better).

Dataset	**Algorithm**	**FID score (↓)**		
		$s = 10$	$s = 30$	$s = 50$
MNIST	Gen-RKM	**89.825**	**130.497**	**131.696**
	VAE	250	234.749	205.282
	β-TCVAE	221.45	182.93	158.31
	InfoGAN	238.75	204.63	179.43
CelebA	Gen-RKM	**103.299**	**84.403**	**85.121**
	VAE	286.039	245.738	225.783
	β-TCVAE	248.47	226.75	173.21
	InfoGAN	264.79	228.31	185.93
fMNIST	Gen-RKM	**93.437**	**127.893**	146.643
	VAE	239.492	211.482	196.794
	β-TCVAE	206.784	187.221	**136.466**
	InfoGAN	247.853	215.683	199.337
CIFAR10	Gen-RKM	**122.475**	**138.467**	**158.871**
	VAE	295.382	259.557	231.475
	β-TCVAE	283.58	214.681	168.483
	InfoGAN	295.321	258.471	220.482

implementation. After trying various values we noticed that $\beta = 3$ seemed to work well across all datasets that we considered. For a fair comparison, the models have the same encoder/decoder architecture, optimization parameters and are trained until convergence, where the details are given in Table E.3.2. We evaluate the performance qualitatively by comparing reconstruction and random sampling, the results are shown in Figure E.4.5 in the Appendix. In order to quantitatively assess the quality of the randomly generated samples, we use the Fréchet Inception Distance (FID) introduced by [98]. The results are reported in Table 5.2.1. Experiments were repeated for different latent-space dimensions (s), and we observe empirically that FID scores are better for the Gen-RKM. This is confirmed by the qualitative evaluation in Figure E.4.5. An interesting trend could be noted that as the dimension of latent-space is increased, VAE gets better at generating images whereas the performance of Gen-RKM decreases slightly. This is attributed to the eigendecomposition of the kernel matrix whose eigenvalue spectrum decreases rapidly depicting that most information is captured in few principal components, while the rest is noise. The presence of noise hinders the convergence of the model. It is therefore important to select the number of latent variables proportionally to the size of the mini-batch and the corresponding spectrum of the kernel matrix (the diversity within a mini-batch affects the eigenvalue spectrum of the kernel matrix).

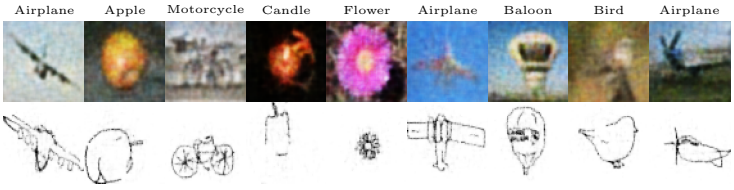


Figure 5.2.5: Multi-view generation on Sketchy dataset showing labels, images and sketches generated together from the common subspace.

Multi-view generation. Figures 5.2.5 & 5.2.6 demonstrate the multi-view generative capabilities of the model. In these datasets, labels or attributes are seen as another view of the image that provides extra information. One-hot encoding of the labels was used to train the model. The first figure of Figure 5.2.6 shows the generated images and labels when feature maps are only implicitly known i.e. through a Gaussian kernel. The second and third figures of Figure 5.2.6 show the same when using fully-connected networks as parametric functions to encode and decode labels. Next, we show an illustration of multi-view generation on the Sketchy database [205]. The dataset is a collection of sketch-photo pairs resulting in 3 views: images, sketches, and labels. The dataset includes 125 object categories with 12,500 natural object images and 75,471 hand-drawn sketches for each class. The following pre-processing is done before training the GEN-RKM model: for the sketchy dataset, we selected 10 classes for training: airplane, apple, bicycle, candle, door, flower, hot-air-balloon, motorcycle, songbird, and teapot. After that, the images are resized to $64 \times 64 \times 3$. Further details on the used model architectures and hyperparameters are given in Appendix E.3. The learned latent space is visualized in Figure 5.2.4. One can clearly observe that the joint learning of the different views results in a better separation of the classes. Joint random generations are given in Figure 5.2.5.

Disentanglement. 1) Qualitative examples: The latent variables are uncorrelated, which gives an indication that the model could resemble a disentangled representation. This is confirmed by the empirical evidence in Figure 5.2.7, where we explore the uncorrelated features learned by the models on the Dsprites and celebA datasets. In our experiments, the Dsprites training dataset comprised of 32×32 positions of oval and heart-shaped objects. The number of principal components chosen was 2 and the goal was to find out whether traversing in the direction of principal components, corresponds to traversing the generated images in one particular direction while preserving the shape of the object. Rows 1 and 2 of Figure 5.2.7 show the reconstructed images of an

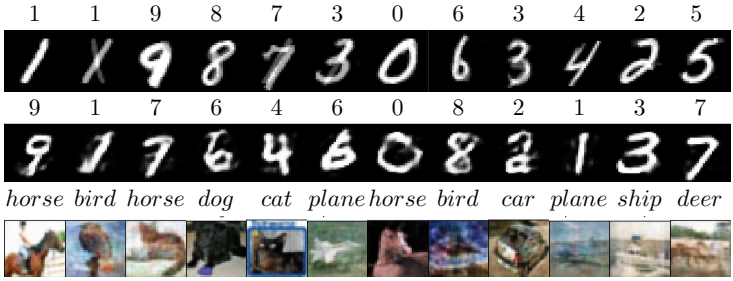


Figure 5.2.6: Multi-view Generation (images and labels) on various datasets using implicit and explicit feature maps. a) MNIST: Implicit feature maps with Gaussian kernel are used during training. For generation, the pre-images are computed using the kernel-smoother method. b, c) MNIST and CIFAR-10: Explicit feature maps and the corresponding pre-image maps are defined by the Convolutional Neural Networks and Transposed CNNs respectively.

oval while moving along the first and second principal components respectively. Notice that the first and second components correspond to the y and x positions respectively. Rows 3 and 4 show the same for hearts. On the celebA dataset, we train the Gen-RKM with 15 components on a subset. Rows 5 and 6 show the reconstructed images while traversing along the principal components. When moving along the first component from left to right, the hair color of the women changes, while preserving the face structure. Whereas traversal along the second component transforms a man to a woman while preserving the orientation. When the number of principal components was 2 while training, the brightness and background light source correspond to the two largest variances in the dataset. Also, notice that the reconstructed images are more blurry due to the selection of fewer components to model $\mathcal{H}$.

2) Quantitative comparisons: To quantitatively assess disentanglement performance, we compare Gen-RKM with VAE [122] and β-VAE [99] on the Dsprites and Teapot datasets [72]. The models have the same encoder/decoder architecture, optimization parameters and are trained until convergence, where the details are given in Table E.3.2. The performance is measured using the proposed framework[3] of [72], which gives 3 measures: disentanglement, completeness and informativeness. The results are shown in Table 5.2.2. Gen-RKM has good performance on the Dsprites dataset when the latent space dimension is equal to 2. This is expected as the number of disentangled generating factors in the dataset is also equal to 2, hence there are no noisy components in the kernel PCA hindering the convergence. The opposite happens

[3]Code and dataset available at `https://github.com/cianeastwood/qedr`

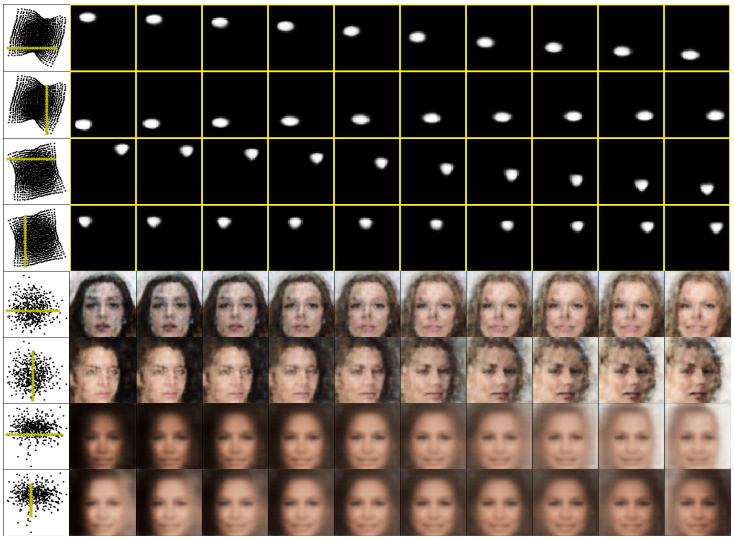


Figure 5.2.7: Exploring the learned uncorrelated features by traversing along the eigenvectors. The first column shows the scatter plot of latent variables using the top two principal components. The green lines within, show the traversal in the latent space and the related rows show the corresponding reconstructed images. The faces shown above are fake.

in the case $h_{dim} = 10$, where noisy components are present. The above is confirmed by the Relative Importance Matrix on Figure E.4.1 in the Appendix, where the 2 generating factors are well separated in the latent space of the Gen-RKM. For the Teapot dataset, Gen-RKM has good performance when $s = 10$. More components are needed to capture all variations in the dataset, where the number of generating factors is now equal to 5. In the other cases, Gen-RKM has a performance comparable to the others. Note that the model selection was done a-priori, that is, the hyperparameters of classifiers were selected before evaluating the disentanglement metric. This may explain the poor scores for Gen-RKM with Random Forest classifier in Teapot dataset ($s = 5$).

Table 5.2.2: Disentanglement Metric on DSprites and Teapot dataset with Lasso and Random Forest regressor [72]. For disentanglement and completeness higher score is better, for informativeness, lower is better.

	s	Algorithm	Lasso			Random Forest		
			Dis. (↑)	**Com.** (↑)	**Inf.** (↓)	**Dis.** (↑)	**Com.** (↑)	**Inf.** (↓)
DSprites	10	Gen-RKM	0.30	0.10	0.87	0.12	0.10	0.28
		VAE	0.11	0.09	**0.17**	**0.73**	**0.54**	**0.06**
		β-VAE ($\beta = 3$)	0.53	**0.18**	0.18	0.58	0.36	**0.06**
		β-TCVAE ($\beta = 3$)	**0.55**	0.17	0.18	0.72	**0.54**	0.11
		Info-GAN	0.37	0.13	0.22	0.61	0.35	0.15
	2	Gen-RKM	**0.72**	**0.71**	**0.64**	**0.05**	0.19	**0.03**
		VAE	0.04	0.01	0.87	0.01	0.13	0.11
		β-VAE ($\beta = 3$)	0.13	0.40	0.71	0.00	**0.26**	0.09
		β-TCVAE ($\beta = 3$)	0.51	0.15	0.67	0.03	0.17	0.14
		Info-GAN	0.46	0.14	0.66	0.04	0.17	0.21
Teapot	10	Gen-RKM	0.28	0.23	0.39	**0.48**	**0.39**	**0.19**
		VAE	0.28	0.21	**0.36**	0.30	0.27	0.21
		β-VAE ($\beta = 3$)	0.33	**0.25**	**0.36**	0.31	0.24	0.20
		β-TCVAE ($\beta = 3$)	**0.35**	0.24	0.39	0.35	0.25	0.31
		Info-GAN	0.23	0.2	0.41	0.32	0.21	0.22
	5	Gen-RKM	0.22	0.23	0.74	0.08	0.09	**0.27**
		VAE	0.16	0.14	**0.66**	0.11	0.14	0.28
		β-VAE ($\beta = 3$)	0.31	0.25	0.68	**0.13**	0.15	0.29
		β-TCVAE ($\beta = 3$)	**0.33**	**0.26**	0.69	0.12	**0.16**	0.29
		Info-GAN	0.21	0.19	0.71	0.11	0.14	0.28

5.3 Disentangled representation learning and generation with manifold optimization

5.3.1 Introduction

In the previous section, the interconnection matrix U is calculated using the eigendecomposition of the kernel matrix. This exact decomposition could lead to numerical difficulties in large-scale datasets. In this section, the interconnection matrix U is restricted to be a rectangular matrix with orthonormal columns, i.e., valued on a Stiefel manifold. The training algorithm now involves a stochastic optimization method using the Cayley ADAM algorithm of [138], which increases only marginally the computing time compared to an analogous VAE. Additionally, we establish connections with probabilistic models, formulate an Evidence Lower Bound (ELBO), and discuss the independence of latent factors.

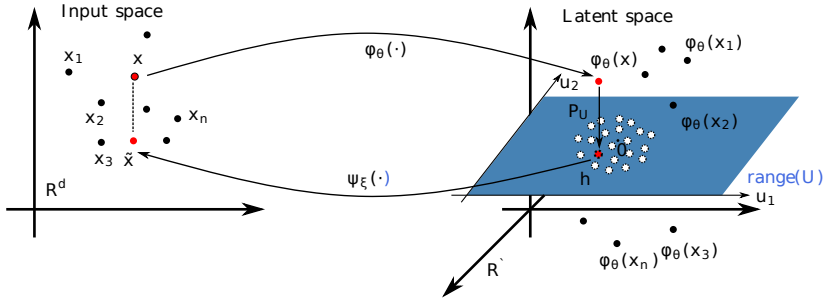


Figure 5.3.1: Schematic illustration of St-RKM training problem. The length of the dashed line represents the reconstruction error (see Auto-Encoder term in (St-RKM)) and the length of the vector projecting on the hyperplane represents the PCA reconstruction error. After training, the projected points tend to be distributed normally on the hyperplane.

Motivation. Let $p(\boldsymbol{x})$ be the distribution of the data $\boldsymbol{x} \in \mathbb{R}^d$ and consider latent vectors $\boldsymbol{z} \in \mathbb{R}^\ell$ with the prior distribution $p(\boldsymbol{z})$, chosen to be a standard normal. Then, one defines an encoder $q(\boldsymbol{z}|\boldsymbol{x})$ that can be deterministic or random, for e.g. given by $\mathcal{N}(\boldsymbol{z}|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), \gamma^2\mathbb{I})$, where the mean[4] is given by the neural network $\phi_{\boldsymbol{\theta}}$ parametrized by $\boldsymbol{\theta}$. A random decoder $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\psi_{\boldsymbol{\xi}}(\boldsymbol{z}), \sigma_0^2\mathbb{I})$ is associated to the decoder neural network $\psi_{\boldsymbol{\xi}}$, parametrized by $\boldsymbol{\xi}$, which maps latent codes to the data points. A VAE is estimated by maximizing the lower bound

$$\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log(p(\boldsymbol{x}|\boldsymbol{z}))] - \beta\,\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x}), p(\boldsymbol{z})) \leq \log p(\boldsymbol{x}), \tag{5.3.1}$$

which is called the Evidence Lower Bound (ELBO) when $\beta = 1$. It has been argued in [99] that larger values of $\beta > 1$ promote more disentanglement. In this section, we attempt to reconcile generation quality with disentanglement. To introduce the model, we firstly make explicit the connection between β-VAEs and standard Auto-Encoders (AEs). Let the dataset be $\{\boldsymbol{x}_i\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$. Let $q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(z|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), \gamma^2\mathbb{I})$ be an encoder, where $\boldsymbol{z} \in \mathbb{R}^\ell$. For a fixed $\gamma > 0$, the maximization problem (5.3.1) is then equivalent to the minimization of the regularized AE

$$\min_{\boldsymbol{\theta},\boldsymbol{\xi}} \frac{1}{n}\sum_{i=1}^{n}\Big\{\mathbb{E}_\epsilon\|\boldsymbol{x}_i - \psi_{\boldsymbol{\xi}}(\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \epsilon)\|_2^2 + \alpha\|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2\Big\}, \tag{5.3.2}$$

where $\alpha = \beta\sigma_0^2$, $\epsilon \sim \mathcal{N}(0, \gamma^2\mathbb{I})$ and where additive constants depending on γ have been omitted. The first term in (5.3.2) might be interpreted as an AE loss

[4]The common definition of a VAE includes another neural network for parametrizing the covariance matrix. To simplify this introductory discussion, this matrix is here chosen as a constant diagonal $\gamma^2\mathbb{I}$.

whereas the second term can be viewed as a regularization. This regularized AE interpretation motivates our method as introduced below.

5.3.2 Proposed model

The optimization problem proposed here is best understood by considering the analogy between Variational Auto-Encoders and classical Auto-Encoders.

Training jointly an auto-encoder and principal component analysis in latent space

The idea consists of learning an auto-encoder along with finding an optimal linear subspace of the latent space so that the variance of the training set in latent space is maximized within this space. See Figure 5.3.1 to follow the discussion below. The encoder $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^\ell$ typically sends input data to a latent space while the decoder $\psi_{\boldsymbol{\xi}} : \mathbb{R}^\ell \to \mathbb{R}^d$ goes in the reverse direction, and constitutes an approximate inverse. Both the encoder and decoder are neural networks parametrized by vectors of $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$. However, it is unclear how to define a parametrization or an architecture of these neural networks so that the learned representation is disentangled. Therefore, in addition to these trained parameters, we also jointly find an m-dimensional linear subspace range(U) of the latent space $\mathbb{R}^\ell$, so that the encoded training points mostly lie within this subspace. This linear subspace is given by the span of the orthonormal columns of the $\ell \times m$ matrix

$$U = \begin{bmatrix} | & & | \\ \boldsymbol{u}_1 & \dots & \boldsymbol{u}_m \\ | & & | \end{bmatrix}.$$

The set of $\ell \times m$ matrices with orthonormal columns with $\ell \geq m$ defines the Stiefel manifold $\mathrm{St}(\ell, m)$. For a reference about Stiefel manifold, we refer to [6]. The idea is then to encode input data into a subspace of the latent space by

$$x \mapsto \mathbb{P}_U \phi_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{u}_1^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{x}) \times \begin{bmatrix} | \\ \boldsymbol{u}_1 \\ | \end{bmatrix} + \cdots + \boldsymbol{u}_m^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{x}) \times \begin{bmatrix} | \\ \boldsymbol{u}_m \\ | \end{bmatrix},$$

where the projector onto range(U) is simply $\mathbb{P}_U = UU^\top$.

The orthonormal vectors $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_m\}$ provides directions associated with different generative factors of our generative model, or in other words the principal directions in latent space match orthogonal directions of variation in the data space (see Figure 5.3.1), in the spirit of Principal Component Analysis

(PCA). This argument is more motivated in Section 5.3.3. We propose to train an objective function which is a *trade-off* between

- an AE loss which promotes parameters such that $\psi_{\boldsymbol{\xi}}(\mathbb{P}_U \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \approx \boldsymbol{x}_i$,
- and, a PCA loss which aims to yield $\mathbb{P}_{U^\perp} \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \approx 0$,

for all $i \in [n]$. The parameters trained are the real entries of $\boldsymbol{\theta}, \boldsymbol{\xi}$ as well as the matrix $U \in \mathrm{St}(\ell, m)$. The model proposed in this section is

$$\min_{\substack{U \in \mathrm{St}(\ell,m) \\ \boldsymbol{\theta},\boldsymbol{\xi}}} \lambda \underbrace{\frac{1}{n}\sum_{i=1}^{n} L_{\boldsymbol{\xi},U}\left(\boldsymbol{x}_i, \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)}_{\text{Auto-Encoder objective}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n} \|\mathbb{P}_{U^\perp}\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2}_{\text{PCA objective}}, \tag{St-RKM}$$

where $\lambda > 0$ is a trade-off parameter. We call this model Stiefel-Restricted Kernel Machines (St-RKM) in view of Section 5.3.2, where more details about Restricted Kernel Machines are given. Though, detailed knowledge of RKM is not needed to understand the content of this section. Hereabove, the AE encoder objective can be chosen as

$$L^{(\sigma)}_{\boldsymbol{\xi},U}(\boldsymbol{x}, \boldsymbol{z}) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\mathbb{I})} \left\| \boldsymbol{x} - \psi_{\boldsymbol{\xi}}\big(\mathbb{P}_U \boldsymbol{z} + \sigma U \epsilon\big) \right\|_2^2, \text{ with } \sigma > 0,$$

in analogy with the VAE objective (5.3.2). In Section 5.3.2, other AE losses are discussed. The basic idea is to combine in (St-RKM) different AE losses with a regularization term which penalizes the feature map in the orthogonal subspace $U^\perp$. The PCA interpretation becomes clear if we introduce the covariance matrix

$$C_{\boldsymbol{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\phi_{\boldsymbol{\theta}}^\top(\boldsymbol{x}_i),$$

where the feature map is assumed to be centered, i.e. $\mathbb{E}_{x \sim p(\boldsymbol{x})}[\phi_{\boldsymbol{\theta}}(\boldsymbol{x})] = 0$. Then, for a given positive integer $m \leq \ell$ the classical PCA problem reads

$$\min_{U \in \mathrm{St}(\ell,m)} \mathrm{Tr}\left(C_{\boldsymbol{\theta}} - \mathbb{P}_U C_{\boldsymbol{\theta}} \mathbb{P}_U\right),$$

as it is explained, for instance, in Section 4.1 of [14]. Clearly, if $\mathbb{P}_U$ is the projector on the m principal components, then the columns of U are the eigenvectors of the covariance matrix[5] and we have the following diagonalization

$$U^\top C_{\boldsymbol{\theta}} U = \mathrm{diag}(\boldsymbol{\lambda}),$$

[5]This follows from Proposition 5.3.1 given hereafter and the fact $\mathrm{Tr}\left(\mathbb{P}_U C_{\boldsymbol{\theta}} \mathbb{P}_U\right) = \mathrm{Tr}\left(U^\top C_{\boldsymbol{\theta}} U\right)$. The latter identity uses the cyclicity of the trace and $U^\top U = \mathbb{I}_m$.

where $\boldsymbol{\lambda}$ is a vector containing the principal values. The PCA objective function can be written as a sum of squared errors as follows

$$\operatorname{Tr}\left(C_{\boldsymbol{\theta}} - \mathbb{P}_U C_{\boldsymbol{\theta}} \mathbb{P}_U\right) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbb{P}_{U^\perp}\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2,$$

where $\mathbb{P}_{U^\perp} = \mathbb{I} - \mathbb{P}_U$. The above PCA objective corresponds to the reconstruction error of Kernel PCA, for the kernel $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \phi_{\boldsymbol{\theta}}^\top(\boldsymbol{x})\phi_{\boldsymbol{\theta}}(\boldsymbol{y})$.

Proposition of two AE losses

Here, we consider two stochastic AE losses. The first loss reads

$$L_{\boldsymbol{\xi},U}^{(\sigma)}(\boldsymbol{x}, \boldsymbol{z}) = \mathbb{E}_\epsilon \left\|\boldsymbol{x} - \psi_{\boldsymbol{\xi}}\left(\mathbb{P}_U \boldsymbol{z} + \sigma U\epsilon\right)\right\|_2^2,$$

where $\epsilon \sim \mathcal{N}(0, \mathbb{I}_m)$. As motivated by Lemma 5.3.1 below, the noise term $\sigma U\epsilon$ above promotes a *smoother* decoder network. To further improve disentanglement, we propose a splitted AE loss

$$L_{\boldsymbol{\xi},U}^{(\sigma),sl}(\boldsymbol{x}, \boldsymbol{z}) = L_{\boldsymbol{\xi},U}^{(0)}(\boldsymbol{x}, \boldsymbol{z}) + \mathbb{E}_\epsilon \left\|\psi_{\boldsymbol{\xi}}\left(\mathbb{P}_U \boldsymbol{z}\right) - \psi_{\boldsymbol{\xi}}\left(\mathbb{P}_U \boldsymbol{z} + \sigma U\epsilon\right)\right\|_2^2, \qquad (5.3.3)$$

with $\epsilon \sim \mathcal{N}(0, \mathbb{I}_m)$. The first term in (5.3.3) is the classical AE loss ($\sigma = 0$) while the second term promotes orthogonal directions of variations, as is discussed further in Section 5.3.3.

In the proposed model, reconstruction of an out-of-sample point x is given by $\psi_{\boldsymbol{\xi}}\big(\mathbb{P}_U \phi_{\boldsymbol{\theta}}(\boldsymbol{x})\big)$. Note that we don't simply propose another encoder-decoder architecture, given by: $U^\top \phi_{\boldsymbol{\theta}}(\cdot)$ and $\psi_{\boldsymbol{\xi}}(U\cdot)$. Instead, our objective assumes that the neural network defining the encoder provides a better embedding if we impose that it maps training points on a linear subspace of dimension $m < \ell$ in the ℓ-dimensional latent space. In other words, the optimization of the parameters in the last layer of the encoder does not play a redundant role, since the second term in (St-RKM) clearly also depends on $\mathbb{P}_{U^\perp}\phi_{\boldsymbol{\theta}}(\cdot)$. The full procedure is described in Algorithm 5.3.1. The problem (St-RKM) was inspired by Restricted Kernel Machines (RKM) by [224], which are briefly described in the next section.

Restricted kernel machines

The above-considered model is based on the variant of Restricted Kernel Machines (RKM) [224] formulation, which can be interpreted as adding the RKM objective function to the AE objective of the form (5.3.2) with $\sigma \approx 0$. RKMs

Table 5.3.1: FID Scores [98] for 8000 randomly generated samples (smaller is better). St-RKM variants are shaded and outperform competitors in all datasets but one.

Models	MNIST	fMNIST	SVHN	Dsprites	3Dshapes	Cars3D
St-RKM ($\sigma = 0$)	**28.71** ($\pm$0.33)	67.70 ($\pm$0.50)	62.97 ($\pm$0.34)	88.82 ($\pm$1.32)	25.76 ($\pm$1.74)	174.42 ($\pm$0.32)
St-RKM ($\sigma = 10^{-3}$)	**28.83** ($\pm$0.23)	**66.84** ($\pm$0.28)	**60.42** ($\pm$0.32)	84.91 ($\pm$1.81)	**21.87** ($\pm$0.18)	**169.86** ($\pm$0.44)
St-RKM-sl ($\sigma = 10^{-3}$)	**28.58** ($\pm$0.21)	73.85 ($\pm$0.36)	**60.40** ($\pm$0.34)	75.94 ($\pm$0.82)	23.14 ($\pm$0.38)	174.76 ($\pm$ 0.52)
VAE ($\beta = 1$)	39.38 ($\pm$0.31)	101.26 ($\pm$0.54)	71.13 ($\pm$0.36)	119.55 ($\pm$1.46)	37.62 ($\pm$1.63)	213.09 ($\pm$0.30)
β-VAE ($\beta = 3$)	30.14 ($\pm$0.19)	86.12 ($\pm$0.62)	72.93 ($\pm$0.47)	83.25 ($\pm$1.87)	30.39 ($\pm$1.01)	172.39 ($\pm$0.41)
FactorVAE	35.12 ($\pm$1.32)	91.43 ($\pm$2.16)	87.45 ($\pm$1.4)	**61.83** ($\pm$1.23)	41.45 ($\pm$1.66)	175.21 ($\pm$0.22)
Info-GAN	77.75 ($\pm$2.05)	78.77 ($\pm$12.51)	98.10 ($\pm$1.21)	121.46 ($\pm$2.84)	55.11 ($\pm$3.18)	177.14 ($\pm$ 0.21)

yield a representation of kernel methods with visible and hidden units; thereby establishing links between Kernel Principal Component Analysis (KPCA) and RBMs. This framework has an energy form similar to RBMs [133] and there is a training procedure in a non-probabilistic setting. The optimization problem (St-RKM) can be expressed as the sum of a regularized AE loss and RKM as follows

$$\min_{\substack{U\in \mathrm{St}(\ell,m)\\ \boldsymbol{\theta},\boldsymbol{\xi}}} \min_{\boldsymbol{h}_i\in\mathbb{R}^m} \sum_{i=1}^{n} \Big\{ \frac{\lambda}{2} L_{\boldsymbol{\xi},U}(\boldsymbol{x}_i, \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \underbrace{f(\boldsymbol{h}_i) - \phi_{\boldsymbol{\theta}}^{\top}(\boldsymbol{x}_i) U \boldsymbol{h}_i}_{\text{RKM}} + \frac{1}{2}\|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2 \Big\},$$

where $\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \in \mathbb{R}^{\ell}$, $\boldsymbol{h}_i i \in \mathbb{R}^m$ with $m \leq \ell$ and U is an interconnection matrix[6]. The function $f : \mathbb{R}^m \to]-\infty, +\infty]$ is used for regularization and for instance can be chosen as closed and strongly convex, or as the characteristic function of a closed set. The analogy with RBM goes as follows: $\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ is interpreted as visible 'units' whereas U plays the role of an interconnection matrix with hidden features $\boldsymbol{h}_i$, which are not binary-valued contrary to RBMs. The first minimization in the above problem gives $\boldsymbol{h}_i^{\star} = U^{\top}\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$, which yields the first term of the optimal objective $-f^{\star}(U^{\top}\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$ in terms of $f^{\star}$, i.e., the Fenchel conjugate of f. In this section, we focus mainly on the squared norm regularizer $f(h) = \frac{1}{2}\|\boldsymbol{h}\|_2^2$, which yields our main objective (St-RKM).

5.3.3 PCA and disentanglement

Decoder smoothness and disentanglement

In the case of the stochastic loss, the smoothness of the decoder is motivated by the following Lemma which extends the result of [191]. Here we adapt it in the context of optimization on the Stiefel manifold. See Appendix F.1.1 for the proof.

[6] In [224], the constraint on U is implemented as a soft penalty term.

Lemma 5.3.1. *Let $\epsilon \sim \mathcal{N}(0, \mathbb{I}_m)$ a random vector and $U \in \mathrm{St}(\ell, m)$. Let $\psi_a(\cdot) \in \mathcal{C}^2(\mathbb{R}^\ell)$ with $a \in [d]$. If the function $[\psi(\cdot) - x]_a^2$ has L_a-Lipschitz continuous Hessian for all $a \in [d]$, we have*

$$\mathbb{E}_\epsilon[\boldsymbol{x} - \psi(\boldsymbol{y} + \sigma U\epsilon)]_a^2 = [\boldsymbol{x} - \psi(\boldsymbol{y})]_a^2 + \sigma^2 \operatorname{Tr}\left(U^\top \nabla\psi_a(\boldsymbol{y}) \nabla\psi_a(\boldsymbol{y})^\top U\right) \quad (5.3.4)$$

$$- \sigma^2 [\boldsymbol{x} - \psi(\boldsymbol{y})]_a \operatorname{Tr}\left(U^\top \operatorname{Hess}_y[\psi_a] U\right) + R_a(\sigma),$$

with $|R_a(\sigma)| \leq \frac{1}{6}\sigma^3 L_a \frac{\sqrt{2}(m+1)\Gamma((m+1)/2)}{\Gamma(m/2)}$ where Γ is Euler's Gamma function.

In Lemma 5.3.1, the additional terms proportional to σ^2 can be interpreted as biases. Specifically, the second term on the right-hand side of the equation in Lemma 5.3.1 indicates that the stochastic AE loss promotes a smooth decoder by penalizing its derivative.

Remark 5.3.1. *For a classical neural network architecture, it is unclear in practice if the function $[\psi(\cdot) - x]_a^2$ has L_a-Lipschitz continuous Hessian for all $a \in [d]$. This assumption in Lemma 5.3.1 is used to provide a meaningful bound on the reminder in* (5.3.4).

Disentanglement

Here we argue that *the principal directions in latent space match orthogonal directions of variation in the data space.* Therefore, the disentanglement of our representation is due to the optimization over $U \in \mathrm{St}(\ell, m)$ and is promoted by the stochastic AE loss. Let $\Delta_k = \nabla\psi(\boldsymbol{y})^\top \boldsymbol{u}_k$ with $1 \leq k \leq m$ and $t \in \mathbb{R}$. Denote by Δ the matrix with Δ_k as columns. Then, as one moves from a point y in the latent space in the direction of $\boldsymbol{u}_k$, the generated data changes by

$$\psi(\boldsymbol{y} + t\boldsymbol{u}_k) - \psi(\boldsymbol{y}) = t\Delta_k + \mathcal{O}(t^2).$$

Consider now a different direction, i.e., $k \neq k'$. A disentangled representation would satisfy: $\Delta_k^\top \Delta_{k'} = 0$ since $\boldsymbol{u}_k$ and $\boldsymbol{u}_{k'}$ are orthogonal. In other words, as the latent point moves along $\boldsymbol{u}_k$ or along $\boldsymbol{u}_{k'}$, the decoder output varies in a significantly different manner. *Hence, for all y in the latent space, we expect the Gram matrix $\Delta^\top\Delta$ to be diagonal.*

To isolate the connection between the equation in Lemma 5.3.1 and the disentanglement of the representation, we assume that the encoder is already mapping points in the subspace defined by U, i.e., $\boldsymbol{y}_i = UU^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \approx \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ for all $1 \leq i \leq n$. From (5.3.4), we observe that the stochastic AE objective includes diagonalization terms involving the trace of a symmetric matrix. Thanks to the Proposition 5.3.1 below, if we consider only the second term in (5.3.4) and

take $M_i = \nabla\psi(\boldsymbol{y}_i)\nabla\psi(\boldsymbol{y}_i)^\top$, we see that the optimization over $U \in \mathrm{St}(\ell, m)$ promotes a diagonal Gram matrix $\Delta^\top\Delta = U^\top M_i U$.

Proposition 5.3.1. *Let M be a $\ell\times\ell$ symmetric matrix with distinct eigenvalues. Let $\nu_1, \ldots, \nu_m$ be its m smallest eigenvalues, with the associated eigenvectors $v_1, \ldots, v_m$. Let V be the matrix whose columns are these eigenvectors. Then, the optimization problem $\min_{U\in\mathrm{St}(\ell,m)} \mathrm{Tr}(U^\top MU)$ is solved by $U^\star = V$ and we have ${U^\star}^\top MU^\star = \mathrm{diag}(\boldsymbol{\nu})$, with $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_m)^\top$.*

5.3.4 Connections with the Evidence Lower Bound

We now discuss the interpretation of the proposed model in the probabilistic setting and the independence of latent factors. In order to formulate an ELBO, consider the following random encoders:

$$q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), \gamma^2\mathbb{I}_\ell) \text{ and } q_U(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mathbb{P}_U\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2\mathbb{P}_U + \delta^2\mathbb{P}_{U^\perp}),$$

where $\phi_{\boldsymbol{\theta}}$ has zero mean on the data distribution. Here, σ^2 plays the role of a trade-off parameter, while the regularization parameter δ is introduced for technical reasons and is put to a numerically small absolute value (see Appendix F.1.2 material for details). Let the decoder be $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(x|\psi_{\boldsymbol{\xi}}(\boldsymbol{z}), \sigma_0^2\mathbb{I})$ and the latent space distribution is parametrized by $p(\boldsymbol{z}) = \mathcal{N}(0, \Sigma)$ where $\Sigma \in \mathbb{R}^{\ell\times\ell}$ is a covariance matrix. We treat Σ as a parameter of the optimization problem that is determined at the last stage of the training. Then the minimization problem (St-RKM) with stochastic AE loss is equivalent to the maximization of

$$\frac{1}{n}\sum_{i=1}^{n}\Big\{\underbrace{\mathbb{E}_{q_U(\boldsymbol{z}|\boldsymbol{x}_i)}[\log(p(\boldsymbol{x}_i|\boldsymbol{z}))]}_{\text{(I)}} - \underbrace{\mathrm{KL}(q_U(\boldsymbol{z}|\boldsymbol{x}_i), q(\boldsymbol{z}|\boldsymbol{x}_i))}_{\text{(II)}} - \underbrace{\mathrm{KL}(q_U(\boldsymbol{z}|\boldsymbol{x}_i), p(\boldsymbol{z}))}_{\text{(III)}}\Big\}, \tag{5.3.5}$$

which is a lower bound to the ELBO, since the KL divergence in (II) is positive. For details of the derivation, see Appendix F.1.2. The hyper-parameters γ, σ, σ_0 take a fixed value. Up to additive constants, the terms (I) and (II) of (5.3.5) match the objective (St-RKM). The third term (III) in (5.3.5) is optimized after the training of the first two terms. It can be written as follows

$$\frac{1}{n}\sum_{i=1}^{n}\mathrm{KL}(q_U(\boldsymbol{z}|\boldsymbol{x}_i), p(\boldsymbol{z})) = \frac{1}{2}\mathrm{Tr}[\Sigma_0\Sigma^{-1}] + \frac{1}{2}\log(\det\Sigma) + \text{constants}$$

with $\Sigma_0 = \mathbb{P}_U C_{\boldsymbol{\theta}}\mathbb{P}_U + \sigma^2\mathbb{P}_U + \delta^2\mathbb{P}_{U^\perp}$. Hence, in that case, the optimal covariance matrix is diagonalized $\Sigma = U(\mathrm{diag}(\boldsymbol{\lambda}) + \sigma^2\mathbb{I}_m)U^\top + \delta^2\mathbb{P}_{U_\perp}$, with $\boldsymbol{\lambda}$ denoting the principal values of the PCA.

Now we briefly discuss the factorization of the encoder. Let $h(\boldsymbol{x}) = U^\top \phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ and let the 'effective' latent variable be $\boldsymbol{z}^{(U)} = U^\top \boldsymbol{z} \in \mathbb{R}^m$. Then the probability density function of $q_U(\boldsymbol{z}|\boldsymbol{x})$ is

$$f_{q_U(\boldsymbol{z}|\boldsymbol{x})}(\boldsymbol{z}) = \frac{e^{-\frac{\|U_\perp^\top \boldsymbol{z}\|_2^2}{2\delta^2}}}{(\sqrt{2\pi\delta^2})^{\ell-m}} \prod_{j=1}^{m} \frac{e^{-\frac{(\boldsymbol{z}_j^{(U)} - \boldsymbol{h}_i j(\boldsymbol{x}))^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}},$$

where the first factor is approximated by a Dirac delta if $\delta \to 0$. Hence, the factorized form of q_U indicates the independence of the latent variables $z^{(U)}$. This independence has been argued to promote disentanglement. In particular, the term (II) in (5.3.5) is analogous to a 'Total Correlation' loss [42], although not formally equal.

5.3.5 Experiments

In this section, we investigate if St-RKM[7] can simultaneously achieve (i) accurate reconstructions on training data (ii) good random generations, and (iii) good disentanglement performance. We use the standard datasets: MNIST [132], Fashion-MNIST [251] (fMNIST), and SVHN [169]. To evaluate disentanglement, we use datasets with known ground-truth generating factors such as Dsprites [154], 3Dshapes [30], and Cars3D [187]. Further, all tables report average errors with 1 standard deviation over 10 experiments.

Algorithm: We use an alternating-minimization scheme (further described in Appendix F.2). First, the Adam optimizer with learning rate 2×10^{-4} is used to update the encoder-decoder parameters and then, the Cayley Adam optimizer [138] with learning rate 10^{-4} is used to update the U. Finally, at the end of the training, we recompute U from the Singular Value Decomposition (SVD) of the covariance matrix as a final correction-step of the Kernel PCA term in our objective (step 10 of the algorithm 5.3.1). Since the $\ell \times \ell$ covariance matrix is typically small, this decomposition is fast (see table F.2.2). In practice, our training procedure only marginally increases the computation cost which can be seen from training times in Table 5.3.2.

Experimental setup: We consider four baselines for comparison: (i) VAE, (ii) β-VAE, (iii) FactorVAE and (iv) Info-GAN. To be consistent in evaluation, we keep the same encoder (discriminator) and decoder (generator) architecture; and the same latent dimension across the models. In the case of Info-GAN, batch-normalization is added for training stability. For the determination of the hyperparameters of other methods, we start from values in the range of

[7] The source code is available at http://bit.ly/StRKM_code

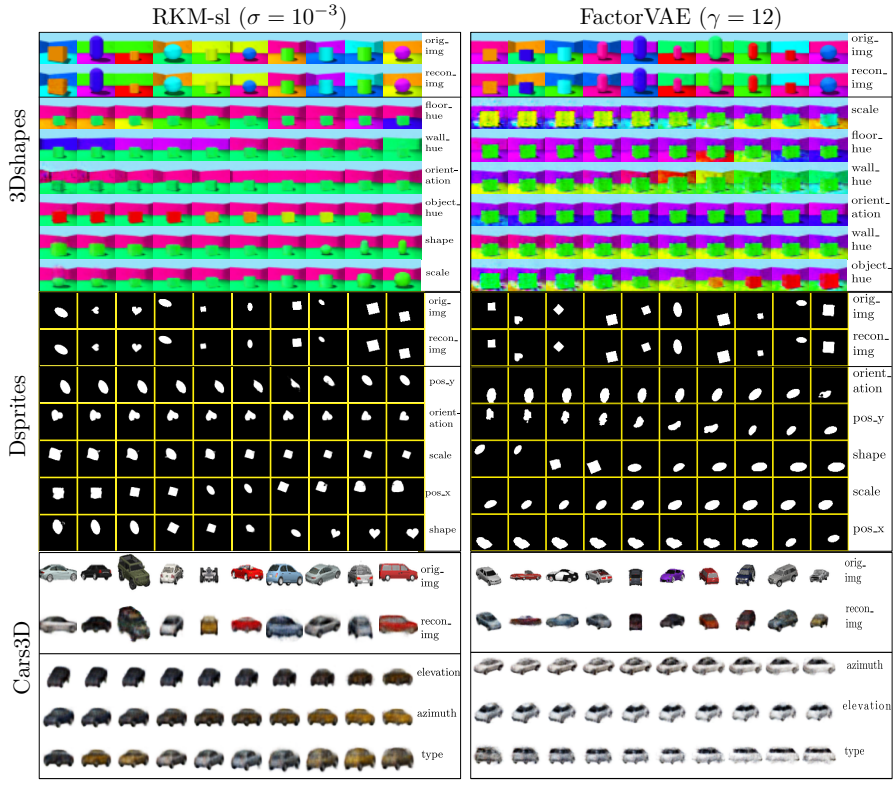


Figure 5.3.2: Traversals along principal components. First-two rows show the ground-truth and reconstructed images. Further each subsequent row show the generated images by traversing along a principal component in the latent space. The last column in each sub-image shows the dominant factor of variation.

Table 5.3.2: Training time in minutes (for 1000 epochs) and number of parameters of the generative models on MNIST dataset.

Model	St-**RKM**	(β)-**VAE**	**FactorVAE**	**Info-GAN**
Nb parameters	4164519	4165589	8182591	4713478
Training time	21.93 ($\pm$1.3)	**19.83** ($\pm$0.8)	33.31 ($\pm$2.7)	45.96 ($\pm$1.6)

the parameters suggested in the authors' reference implementation. After trying various values we noticed that $\beta = 3$ and $\gamma = 12$ seem to work well across the datasets that we considered for β-VAE and FactorVAE respectively. Furthermore, in all experiments on St-RKM, we keep reconstruction weight $\lambda = 1$. All models are trained on the entire dataset. Further technical details

Algorithm 5.3.1 Manifold optimization of St-RKM

Input: $\{\boldsymbol{x}_i\}_{i=1}^n$, $\phi_{\boldsymbol{\theta}}, \psi_{\boldsymbol{\xi}}, \mathcal{J} :=$ Eq. (St-RKM)
Output: Learned $\boldsymbol{\theta}, \boldsymbol{\xi}, U$

procedure TRAIN
  while not converged **do**
    $\{\boldsymbol{x}\} \leftarrow$ {Get mini-batch}
    Get embeddings $\phi_{\boldsymbol{\theta}}(\boldsymbol{x}) \leftarrow \boldsymbol{x}$
    Compute centered $C_{\boldsymbol{\theta}}$ ▷ Covariance matrix
    Update $\{\phi_{\boldsymbol{\theta}}, \psi_{\boldsymbol{\xi}}\} \leftarrow$ Adam$(\mathcal{J})$ ▷ Optimization step
    Update $\{U\} \leftarrow$ Cayley_Adam$(\mathcal{J})$ ▷ Optimization step
  end while
  Do steps 4-5 over whole dataset
  $U \leftarrow \text{SVD}(C_{\boldsymbol{\theta}})$
end procedure

are given in Appendix F.2. Note that for the same encoder-decoder network, the St-RKM model has the least number of parameters compared to any VAE variants and Info-GAN (see Table 5.3.2). To evaluate the quality of generated samples, we report the Fréchet Inception Distance [98] (FID) scores in Table 5.3.1 and the Sliced Wasserstein Distance (SWD) [117] scores in Table 5.3.3. Note that FID scores are not necessarily appropriate for Dsprites since this dataset is significantly different from ImageNet on which the Inception network was originally trained. Randomly generated samples are shown in Figure 5.3.3. To generate samples from deterministic St-RKM ($\sigma = 0$), we sample from a fitted normal distribution on the latent embeddings of the dataset (similar to [84]). St-RKM variants (shaded background) perform better on most datasets and within them, the stochastic variants with $\sigma = 10^{-3}$ perform the best. This can be attributed to a better generalization of the decoder network due to the addition of noise-term on latent variables (see Lemma 5.3.1).The training times for St-RKM variants are shorter compared to FactorVAE and Info-GAN due to a significantly smaller number of parameters.

To evaluate the disentanglement performance, various metrics have been proposed. A comprehensive review by Locatello et al. [146] shows that the various disentanglement metrics are correlated albeit with a different degree of correlation across datasets. In this section, we use three measures given in Eastwood's framework [72]: *disentanglement*: the degree to which a representation factorizes the underlying factors of variation, with each variable capturing at most one generative factor; *completeness*: the degree to which each underlying factor is captured by a single code variable; and *informativeness*: the amount of information that a representation captures

Table 5.3.3: Sliced Wasserstein Distance (SWD) evaluates the quality of randomly generated 8000 samples over 10 iterations (smaller is better). The multi-scale statistical similarity between distributions of local image patches drawn from the Laplacian pyramid is evaluated using the SWD. A small Wasserstein distance indicates that the distribution of the patches is similar, thus real and fake images appear similar in both appearance and variation at this spatial resolution. We always show the average SWD to evaluate performance. Scores are multiplied by 10^2 for better readability.

Models	**MNIST**	**fMNIST**	**SVHN**	**3Dshapes**	**DSprites**	**Cars3D**
St-RKM ($\sigma = 0$)	4.80 (±0.13)	**4.71** (±0.14)	4.36 (±0.32)	2.52 (±0.18)	4.54 (±0.64)	3.69 (±1.4)
St-RKM ($\sigma = 10^{-3}$)	4.77 (±0.12)	6.46 (±0.17)	**3.26** (±0.16)	**1.04** (±0.14)	3.72 (±0.58)	3.62 (±1.29)
St-RKM-sl ($\sigma = 10^{-3}$)	**3.11** (±0.10)	5.17 (±0.10)	4.16 (±0.23)	1.20 (±0.19)	**3.13** (±0.54)	4.02 (±1.40)
VAE ($\beta = 1$)	4.85 (±0.48)	5.60 (±0.09)	4.50 (±0.34)	2.06 (±0.13)	5.04 (±0.92)	4.01 (±1.90)
β-VAE ($\beta = 3$)	3.75 (±0.08)	7.16 (±0.28)	4.71 (±0.27)	3.25 (±0.27)	4.85 (±0.68)	4.83 (±0.21)
FactorVAE ($\gamma = 12$)	3.52 (±0.27)	5.12 (±0.01)	3.46 (±0.64)	1.32 (±0.01)	3.24 (±0.02)	**3.47** (±0.07)
InfoGAN	4.08 (±0.27)	5.21 (±1.33)	4.84 (±0.72)	2.33 (±0.36)	5.17 (±0.31)	4.92 (±0.33)

Table 5.3.4: Eastwood framework's [72] disentanglement metric with Lasso and Random Forest regressor. For disentanglement and completeness higher score is better, for informativeness, lower is better. 'Info.' indicates (average) root-mean-square error in predicting $\boldsymbol{z}$. The best scores are in bold. St-RKM variants are shaded and outperform other models except in two instances.

Dataset	**Model**	**Lasso**			**Random Forest**		
		Dise.	**Comp.**	**Info.**	**Dise.**	**Comp.**	**Info.**
Dsprites	St-RKM ($\sigma = 0$)	0.41 (±0.02)	0.45 (±0.01)	1.05 (±0.03)	0.27 (±0.01)	0.62 (±0.01)	0.97 (±0.03)
	St-RKM ($\sigma = 10^{-3}$)	**0.45** (±0.01)	**0.47** (±0.02)	1.05 (±0.01)	0.28 (±0.01)	**0.63** (±0.02)	1.02 (±0.01)
	St-RKM-sl ($\sigma = 10^{-3}$)	0.37 (±0.03)	0.32 (±0.01)	1.07 (±0.02)	**0.35** (±0.02)	0.58 (±0.01)	**0.96** (±0.02)
	VAE ($\beta = 1$)	0.26 (±0.06)	0.22 (±0.00)	0.97 (±0.01)	0.24 (±0.03)	0.55 (±0.04)	1.00 (±0.01)
	β-VAE ($\beta = 3$)	0.36 (±0.02)	0.31 (±0.02)	0.96 (±0.21)	0.33 (±0.01)	0.53 (±0.04)	0.99 (±0.11)
	FactorVAE ($\gamma = 12$)	0.40 (±0.01)	0.34 (±0.01)	0.98 (±0.01)	0.34 (±0.02)	0.58 (±0.01)	1.05 (±0.01)
	Info-GAN	0.31 (±0.21)	0.27 (±0.03)	**0.95** (±0.02)	0.31 (±0.01)	0.47 (±0.20)	1.00 (±0.02)
3Dshapes	St-RKM ($\sigma = 0$)	**0.76** (±0.02)	**0.71** (±0.02)	1.06 (±0.03)	0.55 (±0.03)	**0.69** (±0.02)	**0.51** (±0.21)
	St-RKM ($\sigma = 10^{-3}$)	0.74 (±0.02)	0.66 (±0.01)	1.24 (±0.02)	0.61 (±0.01)	0.67 (±0.01)	0.86 (±0.10)
	St-RKM-sl ($\sigma = 10^{-3}$)	0.72 (±0.01)	0.65 (±0.01)	**1.03** (±0.02)	**0.63** (±0.02)	0.66 (±0.02)	0.95 (±0.01)
	VAE ($\beta = 1$)	0.44 (±0.21)	0.33 (±0.22)	1.26 (±0.20)	0.33 (±0.20)	0.36 (±0.21)	0.94 (±0.01)
	β-VAE ($\beta = 3$)	0.55 (±0.01)	0.54 (±0.01)	1.07 (±0.01)	0.56 (±0.01)	0.57 (±0.03)	0.54 (±0.22)
	FactorVAE ($\gamma = 12$)	0.62 (±0.01)	0.41 (±0.03)	1.05 (±0.01)	0.57 (±0.02)	0.58 (±0.01)	0.93 (±0.20)
	Info-GAN	0.41 (±0.22)	0.39 (±0.01)	1.17 (±0.02)	0.53 (±0.01)	0.51 (±0.10)	0.61 (±0.12)
Cars3D	St-RKM ($\sigma = 0$)	0.45 (±0.01)	0.27 (±0.13)	1.33 (±0.08)	0.49 (±0.01)	**0.38** (±0.01)	**1.16** (±0.03)
	St-RKM ($\sigma = 10^{-3}$)	0.42 (±0.09)	0.40 (±0.02)	1.34 (±0.03)	0.54 (±0.01)	0.32 (±0.02)	1.20 (±0.11)
	St-RKM-sl ($\sigma = 10^{-3}$)	**0.65** (±0.02)	**0.48** (±0.01)	1.30 (±0.05)	**0.55** (±0.02)	0.33 (±0.02)	1.20 (±0.03)
	VAE ($\beta = 1$)	0.47 (±0.01)	0.18 (±0.04)	1.34 (±0.02)	0.23 (±0.21)	0.35 (±0.01)	1.21 (±0.02)
	β-VAE ($\beta = 3$)	0.51 (±0.06)	0.27 (±0.08)	1.35 (±0.01)	0.47 (±0.07)	0.37 (±0.02)	1.19 (±0.07)
	FactorVAE ($\gamma = 12$)	0.54 (±0.02)	0.38 (±0.23)	1.33 (±0.02)	0.44 (±0.01)	0.33 (±0.01)	1.24 (±0.01)
	Info-GAN	0.56 (±0.01)	0.23 (±0.13)	**1.29** (±0.04)	0.27 (±0.22)	0.32 (±0.05)	1.41 (±0.21)

about the underlying factors of variation. Table 5.3.4 shows that St-RKM variants (shaded background) have better disentanglement and completeness scores. However, informativeness scores are higher for St-RKM when using a lasso-regressor in contrast to mixed scores with Random forest regressor. This can be seen clearly in Figure 5.3.2 which shows the generated images by traversing along the principal components in the latent space. In 3Dshapes dataset, St-RKM model captures floor-hue, wall-hue, and orientation perfectly but has a slight entanglement in capturing other factors. This is worse in β-VAE which has entanglement in all dimensions except the floor-hue along with noise in some generated images. Similar trends can be observed in the Dsprites and Cars3D datasets.

5.4 Conclusion

In Section 5.1, the use of an implicit feature map, e.g the Gaussian kernel, in the generative RKM framework is demonstrated. By using the eigenvalue decomposition for training, the interconnection matrix is orthogonal, which has great benefits towards explainability when exploring the latent space. This is shown on a number of real-life datasets. In Section 5.2, the use of explicit feature maps, in the form of neural networks, is introduced in the GEN-RKM algorithm. We proposed a novel training procedure that jointly learns the features and a shared subspace representation. Next to that, we extended the RKM framework to multi-view generation. Finally, in Section 5.3, we proposed two main changes with respect to the Gen-RKM algorithm, the interconnection matrix U is restricted to be a rectangular matrix with orthonormal columns, i.e., valued on a Stiefel manifold, and for the training, we use the Cayley Adam algorithm of [138] for stochastic optimization on the Stiefel manifold.

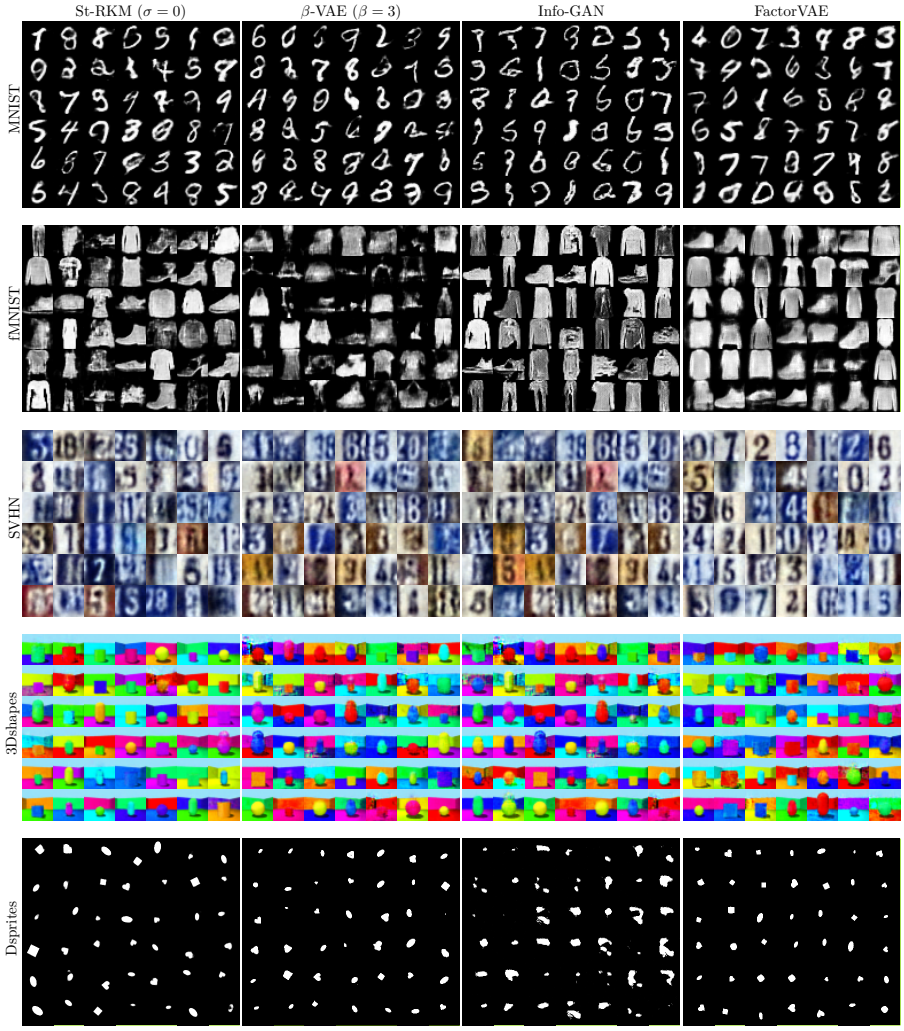


Figure 5.3.3: Samples of randomly generated batch of images used to compute FID scores (see Table 1 in main part) and SWD scores (see Table 5.3.3).

Chapter 6

Robustness

This chapter comprises the previously published articles:

- Arun Pandey, Joachim Schreurs[1], and Johan AK Suykens. Robust generative restricted kernel machines using weighted conjugate feature duality. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 613–624. Springer, 2020
- Joachim Schreurs[2], Iwein Vranckx, Bart De Ketelaere, Mia Hubert, Johan AK Suykens, and Peter J. Rousseeuw. Outlier detection in non-elliptical data by kernel MRCD. In *Statistics and Computing*, 2021.

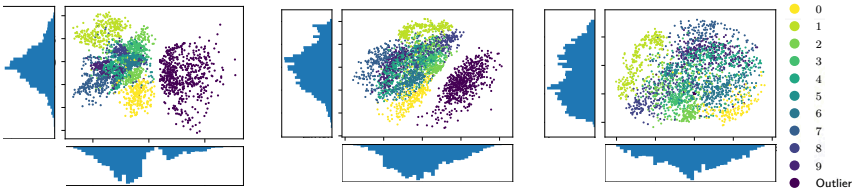
6.1 Robust generative restricted kernel machines using weighted conjugate feature duality

6.1.1 Introduction

In generative modeling, training data is often assumed to be ground truth, therefore outliers can severely degrade the learned representations and performance of trained models. The same issue arises in generative modeling

[1]The candidate wrote the article and defined the methodology in collaboration with the coauthors.

[2]The candidate wrote the article, performed experiments, did the theoretical derivations, and defined the methodology in collaboration with the coauthors.



(a) VAE (b) Gen-RKM (c) Robust Gen-RKM

Figure 6.1.1: Illustration of robustness against outliers on the MNIST dataset. 20% of the training data is contaminated with noise. The models are trained with a 2-dimensional latent space in the standard setup, see Section 6.1.4. The presence of outliers distorts the distribution of the latent variables for the Gen-RKM and VAE, where the histogram of the latent variables is skewed. By down-weighting the outliers, the histogram resembles a Gaussian distribution again.

where contamination of the training data results in the encoding of the outliers. Consequently, the network generates noisy images when reconstructing out-of-sample extensions. To solve this problem, multiple robust variants of generative models were proposed in [79, 45, 227]. However, these generative models require clean training data or only consider the case where there is label noise. In this section, we address the problem of *contamination on the training data itself*. This is a common problem in real-life datasets, which are often contaminated by human error, measurement errors, or changes in system behavior. To the best of our knowledge, this specific problem is not addressed in other generative methods. The Restricted Kernel Machine (RKM) formulation allows for straightforward integration of methods from classical robust statistics to the RKM framework.

We introduce a weighted Gen-RKM model that detects and penalizes the outliers to regularize the latent space. Thanks to the introduction of weighted conjugate feature duality, an RKM formulation for weighted kernel PCA is derived. This formulation is used within the Gen-RKM training procedure to fine-tune the latent space using different weighting schemes. A weighting function based on Minimum Covariance Determinant (MCD) [199] is proposed. Qualitative and quantitative experiments on standard datasets show that the proposed model is unaffected by large contamination and can learn meaningful representations.

6.1.2 Weighted restricted kernel machines

Weighted conjugate feature duality

In this section, we extend the notion of conjugate feature duality by introducing a weighting matrix. Assuming $D \succ 0$ to be a positive-definite diagonal weighting matrix, the following holds for any two vectors $\boldsymbol{e}, \boldsymbol{h} \in \mathbb{R}^n$, $\lambda > 0$:

$$\frac{1}{2\lambda}\boldsymbol{e}^\top D\boldsymbol{e} + \frac{\lambda}{2}\boldsymbol{h}^\top D^{-1}\boldsymbol{h} \geq \boldsymbol{e}^\top \boldsymbol{h}. \tag{6.1.1}$$

The inequality could be verified using the Schur complement by writing the above in its quadratic form:

$$\frac{1}{2}\begin{bmatrix}\boldsymbol{e}^\top & \boldsymbol{h}^\top\end{bmatrix}\begin{bmatrix}\frac{1}{\lambda}DI & -I \\ -I & \lambda D^{-1}I\end{bmatrix}\begin{bmatrix}\boldsymbol{e} \\ \boldsymbol{h}\end{bmatrix} \geq 0. \tag{6.1.2}$$

It states that for a matrix $Q = \left[\begin{smallmatrix} A & B \\ B^\top & C\end{smallmatrix}\right]$, one has $Q \succeq 0$ if and only if $A \succ 0$ and the Schur complement $C - B^\top A^{-1} B \succeq 0$ [26], which proves the above inequality. This is also known as the Fenchel-Young inequality for quadratic functions [190].

We assume a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, with $\boldsymbol{x}_i \in \mathbb{R}^d$ consisting of N data points. For a feature-map $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d_f}$ defined on input data points, the weighted kernel PCA objective [207] in the Least-Squares Support Vector Machine (LS-SVM) setting is given by [11]:

$$\min_{U,\boldsymbol{e}} J(U, \boldsymbol{e}) = \frac{\eta}{2}\operatorname{Tr}(U^\top U) - \frac{1}{2\lambda}\boldsymbol{e}^\top D\boldsymbol{e} \text{ s.t. } \boldsymbol{e}_i = U^\top \phi(\boldsymbol{x}_i), \forall i = 1, \ldots, N, \tag{6.1.3}$$

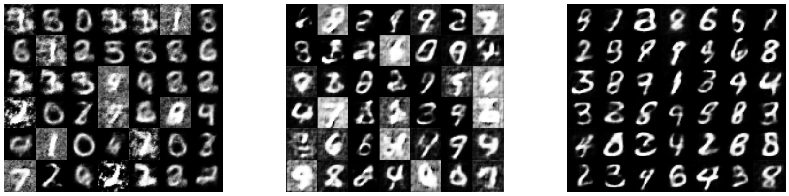
where $U \in \mathbb{R}^{d_f \times s}$ is the unknown interconnection matrix. By using (6.1.1), the error variables $\boldsymbol{e}_i$ are conjugated to latent variables $\boldsymbol{h}_i \in \mathbb{R}^s$ and substituting the constraints into the objective function yields

$$J \leq \mathcal{J}_t^D \coloneqq \sum_{i=1}^N \left\{ -\phi(\boldsymbol{x}_i)^\top U\boldsymbol{h}_i + \frac{\lambda}{2}D_{ii}^{-1}\boldsymbol{h}_i^\top \boldsymbol{h}_i \right\} + \frac{\eta}{2}\operatorname{Tr}(U^\top U). \tag{6.1.4}$$

The stationary points of $\mathcal{J}_t^D$ are given by:

$$\begin{cases} \frac{\partial \mathcal{J}_t^D}{\partial \boldsymbol{h}_i} = 0 \implies \lambda D_{ii}^{-1}\boldsymbol{h}_i = U^\top \phi(\boldsymbol{x}_i), \ \forall i = 1, \ldots, N \\ \frac{\partial \mathcal{J}_t^D}{\partial U} = 0 \implies U = \frac{1}{\eta}\sum_{i=1}^N \phi(\boldsymbol{x}_i)\boldsymbol{h}_i^\top. \end{cases} \tag{6.1.5}$$

Eliminating U and denoting the kernel matrix $K \coloneqq [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{ij}$ with kernel function $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle$, the eigenvectors $h \coloneqq [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N]$, $\Lambda \coloneqq$



(a) VAE (b) Gen-RKM (c) Robust Gen-RKM

Figure 6.1.2: Illustration of robust generation on the MNIST dataset. 20% of the training data is contaminated with noise. The images are generated by random sampling from a fitted Gaussian distribution on the learned latent variables. When using a robust training procedure, the model does not encode the noisy images. As a consequence, no noisy images are generated.

$\text{diag}\{\lambda_1, \ldots, \lambda_s\} \in \mathbb{R}^{s \times s}$ such that $\lambda_1 \geq \ldots \geq \lambda_s$ with s the dimension of the latent space, we get the weighted eigenvalue problem:

$$\frac{1}{\eta}[DK]H^\top = H^\top \Lambda. \tag{6.1.6}$$

One can verify that each eigenvalue-eigenvector pair lead to the value $\mathcal{J}_t^D = 0$. Using the weighted kernel PCA potential outliers can be penalized, which is discussed in more detail in Section 6.1.3.

Generation

Given the learned interconnection matrix $\boldsymbol{U}$, and a latent variable $\boldsymbol{h}^\star$, consider the following objective function

$$\mathcal{J}_g = -\phi(\boldsymbol{x}^\star)^\top U \boldsymbol{h}^\star + \frac{1}{2}\phi(\boldsymbol{x}^\star)^\top \phi(\boldsymbol{x}^\star), \tag{6.1.7}$$

with a regularization term on the input data. Here $\mathcal{J}_g$ denotes the objective function for generation. To reconstruct or denoise a training point, $\boldsymbol{h}^\star$ can be one-of-the corresponding hidden units of the training point. Random generation is done by fitting a normal distribution on the learned latent variables, afterwards a random $\boldsymbol{h}^\star$ is sampled from the distribution which is put through the decoder network. Note that in the training objective (ref. (6.1.4)) we are imposing a soft-Gaussian prior over latent variables through quadratic regularization on $\{\boldsymbol{h}_i\}_{i=1}^N$. The stationary points of (6.1.7) yields the *generated feature vector* $\varphi(\boldsymbol{x}^\star)$, given by the corresponding $\boldsymbol{h}^\star$. With slight abuse of notation, we denote the generated feature-vector by $\varphi(\boldsymbol{x}^\star) = [\frac{1}{\eta}\sum_{i=1}^N \phi(\boldsymbol{x}_i)\boldsymbol{h}_i^\top]\boldsymbol{h}^\star$, which is a point

in the feature-space corresponding to an unknown $\boldsymbol{x}^\star$ in data space. To obtain the generated data in the input space, the inverse image of the feature map $\phi(\cdot)$ should be computed. In kernel methods, this is known as the pre-image problem. We seek to find the function $\psi\colon \mathbb{R}^{d_f} \mapsto \mathbb{R}^d$, such that $(\psi \circ \varphi)(\boldsymbol{x}^\star) \approx \boldsymbol{x}^\star$, where $\varphi(\boldsymbol{x}^\star)$ is calculated from above. The pre-image problem is known to be ill-conditioned [160], and consequently, various approximation techniques have been proposed [104]. Another approach is to explicitly define pre-image maps and learn the parameters in the training procedure as in the Gen-RKM algorithm. In the experiments, we use (convolutional) neural networks as the feature maps $\phi_{\boldsymbol{\theta}}(\cdot)$, where the notation extends to $\varphi_{\boldsymbol{\theta}}(\cdot)$. Another (transposed convolutional) neural network is used for the pre-image map $\psi_{\boldsymbol{\xi}}(\cdot)$ [69]. The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\xi}$ correspond to the network parameters. These parameters are learned by minimizing the reconstruction error in combination with the weighted RKM objective function. The training algorithm is described in more detail in Section 6.1.3.

Remark on Out-of-Sample extension: To reconstruct or denoise an out-of-sample test point $\boldsymbol{x}^\star$, the data is projected on the latent space using:

$$\boldsymbol{h}^\star = \lambda^{-1} U^\top \phi(\boldsymbol{x}^\star) = \frac{1}{\lambda\eta} \sum_{i=1}^{N} \boldsymbol{h}_i k(\boldsymbol{x}_i, \boldsymbol{x}^\star). \tag{6.1.8}$$

The latent point is reconstructed by projecting back to the input space by first computing the generated feature vector followed by its pre-image map $\psi_{\boldsymbol{\xi}}(\cdot)$.

6.1.3 Robust estimation of the latent variables

Robust weighting scheme

In this section, we propose a weighting scheme to make the estimation of the latent variables more robust against contamination. The weighting matrix is a diagonal matrix with a weight D_{ii} corresponding to every $\boldsymbol{h}_i$ such that:

$$D_{ii} = \begin{cases} 1 & \text{if } d_i^2 \leq \chi^2_{s,\alpha} \\ 10^{-4} & \text{otherwise,} \end{cases} \tag{6.1.9}$$

with s the dimension of the latent space, α the significance level of the Chi-squared distribution and d_i^2 the Mahalanobis distance for the corresponding $\boldsymbol{h}_i$:

$$d_i^2 = (\boldsymbol{h}_i - \hat{\boldsymbol{\mu}})^\top \hat{\Sigma}^{-1} (\boldsymbol{h}_i - \hat{\boldsymbol{\mu}}), \tag{6.1.10}$$

with $\hat{\boldsymbol{\mu}}$ and $\hat{\Sigma}$ the robustly estimated mean and covariance matrix respectively. In this section, we propose to use the Minimum Covariance Determinant

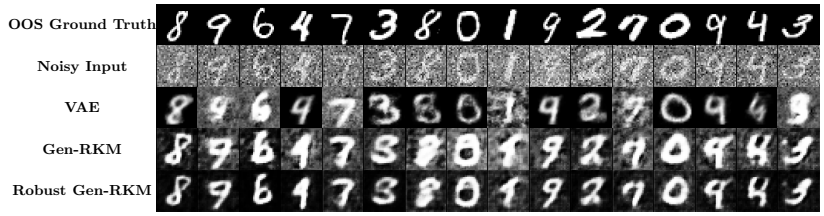


Figure 6.1.3: Illustration of robust denoising on the MNIST dataset. 20 % of the training data is contaminated with noise. The first and second rows show the clean and noisy test images respectively. The third, fourth, and fifth rows show the denoised image using the VAE, Gen-RKM, and robust Gen-RKM respectively.

(MCD) [199]. The MCD is a highly robust estimator of multivariate location and scatter which has been used in many robust multivariate statistical methods [111, 112]. Given a data matrix of N rows with s columns, the objective is to find the $N_{\text{MCD}} < N$ observations whose sample covariance matrix has the lowest determinant. Its influence function is bounded [50] and has the highest possible breakdown value when $N_{\text{MCD}} = \lfloor (N + s + 1)/2 \rfloor$. In the experiments, we typically take $N_{\text{MCD}} = \lfloor N \times 0.75 \rfloor$ and $\alpha = 0.975$ for the Chi-squared distribution. The user could further tune these parameters according to the estimated contamination degree in the dataset. Eventually, the reweighting procedure can be repeated iteratively, but in practice, one single additional weighted step will often be sufficient. Kernel PCA can take the interpretation of a one-class modeling problem with zero target value around which one maximizes the variance [223]. The same holds in the Gen-RKM framework. This is a natural consequence of the regularization term $\frac{\lambda}{2} \sum_{i=1}^{N} \boldsymbol{h}_i^\top \boldsymbol{h}_i$ in the training objective (see (6.1.4)), which implicitly puts a Gaussian prior on the hidden units. When the training of the feature map is done correctly, one expects the latent variables to be normally distributed around zero [225]. Gaussian distributed latent variables are essential for having a *continuous* and *smooth* latent space, allowing easy interpolation. This property is also essential for VAEs and was studied in [122], where a regularization term, in the form of the Kullback-Leibler divergence between the encoder's distribution and a unit Gaussian as a prior on the latent variables was used. When training a non-robust generative model in the presence of outliers, the contamination can severely distort the distribution of the latent variables. This effect is seen in Figure 6.1.1, where a discontinuous and skewed distribution is visible.

Algorithm

We propose to use the above described reweighting step within the Gen-RKM framework. The algorithm is flexible to incorporate both kernel-based, (deep) neural network and Convolutional based models within the same setting, and is capable of jointly learning the feature maps and latent representations. The Gen-RKM algorithm consists out of two phases: a training phase and a generation phase which occurs one after another. In the case of explicit feature maps, the training phase consists of determining the parameters of the explicit feature and pre-image map together with the hidden units $\{\boldsymbol{h}_i\}_{i=1}^N$.

We propose an adapted algorithm of Gen-RKM with an extra re-weighting step wherein the system in (6.1.6) is solved. Furthermore, the reconstruction error is weighted to reduce the effect of potential outliers on the pre-image maps. The loss function now becomes:

$$\min_{\boldsymbol{\theta},\boldsymbol{\xi}} \ \mathcal{J}_c^D(\boldsymbol{\theta},\boldsymbol{\xi}) = \ \mathcal{J}_t^D + \frac{c_{\text{stab}}}{2}(\mathcal{J}_t^D)^2 + \frac{c_{\text{acc}}}{N}\sum_{i=1}^{N} D_{ii}\mathcal{L}(\boldsymbol{x}_i, \psi_{\boldsymbol{\xi}}(\varphi_{\boldsymbol{\theta}}(\boldsymbol{x}_i))), \tag{6.1.11}$$

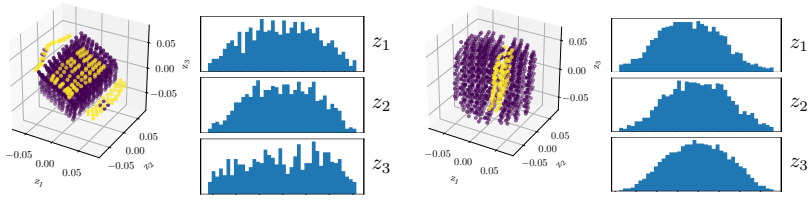
where $c_{stab} \in \mathbb{R}^+$ is a stability constant [224] and $c_{\text{acc}} \in \mathbb{R}^+$ is a regularization constant to control the stability with reconstruction accuracy. In the experiments, the loss function is equal to the mean squared error (MSE), however other loss functions are possible. The generation algorithm is the same as in the Gen-RKM algorithm.

Table 6.1.1: FID Scores [98] over 10 iterations for 4000 randomly generated samples when the training data is contaminated with 20% outliers. (smaller is better).

Dataset	**FID score**			
	VAE	β-**VAE** ($\beta = 3$)	**RKM**	**Rob Gen-RKM**
MNIST	142.54±0.73	187.21±0.11	134.95±1.61	**87.32±1.92**
F-MNIST	245.84±0.43	291.11±1.6	163.51±1.24	**153.32±0.05**
SVHN	168.21±0.23	234.87±1.45	112.45±1.4	**98.14 ±1.2**
CIFAR-10	201.21±0.71	241.23±0.34	187.08±0.58	**132.6±0.21**
Dsprites	234.51±1.10	298.21±1.5	182.65±0.57	**160.56±0.96**
3Dshapes	233.18±0.94	252.41±0.38	177.29±1.60	**131.18±1.45**

6.1.4 Experiments

In this section, we evaluate the robustness of the weighted Gen-RKM on the MNIST, Fashion-MNIST (F-MNIST), CIFAR-10, SVHN, Dsprites and 3Dshapes



(a) Gen-RKM (b) Robust Gen-RKM

Figure 6.1.4: Illustration of disentanglement on the 3DShapes dataset. Clean data is depicted in purple, outliers in yellow. The training subset is contaminated with a third generating factor (20% of the data is considered as outliers). The outliers are down-weighted in the robust Gen-RKM, which moves them to the center.

dataset[1]. The last two datasets will be used in disentanglement experiments since they include the ground truth generating factors that are necessary to quantify the performance. Training of the robust Gen-RKM is done using the algorithm proposed in Section 6.1.3, where we take $N_{\text{MCD}} = \lfloor N \times 0.75 \rfloor$ and $\alpha = 0.975$ for the Chi-squared distribution (see (6.1.9)). Afterwards, we compare with the standard Gen-RKM, VAE, and β-VAE. The models have the same encoder/decoder architecture, optimization parameters and are trained until convergence. Information on the training settings and model architectures is given in Table 6.1.3.

Generation and Denoising: Figure 6.1.2 shows the generation of random images when models were trained on the contaminated MNIST dataset. The contamination consists of adding Gaussian noise $\mathcal{N}(0.5, 0.5)$ to 20% of the data. The images are generated by random sampling from a fitted Gaussian distribution on the learned latent variables. As we can see, when using a robust training procedure, the model does not encode the noisy images. As a consequence, no noisy images are generated and the generation quality is significantly better. This is also confirmed by the Fréchet Inception Distance (FID) scores [98] in Table 6.1.1, which quantifies the quality of generation. The robust Gen-RKM clearly outperforms the other methods when the data has contamination. Moreover, VAEs are known to generate samples closer to the mean of the dataset. This negatively affects the FID scores which also takes into account the diversity within the generated images. The scores for β-VAE are worst due to the inherent emphasis on imposing a Gaussian distribution on

[1] `http://yann.lecun.com/exdb/mnist/`, `https://github.com/zalandoresearch/fashion-mnist`, `https://github.com/deepmind/dsprites-dataset`, `https://github.com/deepmind/3d-shapes`, `https://www.cs.toronto.edu/~kriz/cifar.html`, `http://ufldl.stanford.edu/housenumbers/`

latent variables trading-off with the reconstruction quality [99]. The classical RKM performs slightly better than VAE and its variant. This is attributed to the presence of kernel PCA during training, which is often used in denoising applications and helps to some extent in dealing with contamination in the dataset.

Next, we use generative models in the denoising experiment. Image denoising is accomplished by projecting the noisy test set observations on the latent space, afterwards projecting back to the input space. Because there is a latent bottleneck, the most important features of the images are retained while insignificant features like noise are removed. Figure 6.1.3 shows an illustration of robust denoising on the MNIST dataset. The robust Gen-RKM does not encode the noisy images within the training procedure. Consequently, the model is capable of denoising the out-of-sample test images. When comparing the denoising quality on the full test set (5000 images sampled uniformly at random), the Mean Absolute Error (MAE) of Gen-RKM: MAE = 0.415 and VAE: MAE = 0.434 is much higher than the robust version: MAE = 0.206. The experiments show that basic generative models like Gen-RKM and VAE are highly affected by outliers, while the robust counterpart can cope with a significant fraction of contamination.

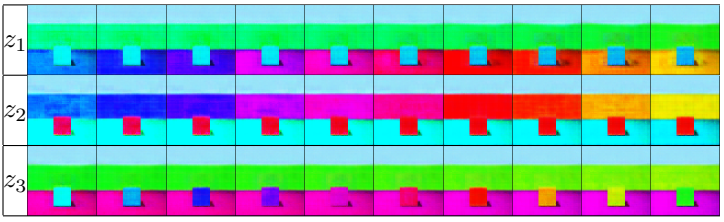


Figure 6.1.5: Illustration of latent traversals along the 3 latent dimensions for 3DShapes dataset using the robust Gen-RKM model. The first, second, and third rows distinctly capture the floor-hue, wall-hue, and object-hue respectively while keeping other generative factors constant.

Effect on disentanglement: In this experiment, contamination is an extra generating factor that is not present in the majority of the data. The goal is to train a disentangled representation, where the robust model only focuses on the most prominent generating factors. We subsample a 'clean' training subset which consists of cubes with different floor, wall, and object hue. The scale and orientation are kept constant with minimum scale and 0° orientation respectively. Afterwards, the training data is contaminated by cylinders with a maximum scale at 30° orientation (20% of the data is considered as outliers).

Table 6.1.2: Disentanglement Metric on DSprites and 3D Shapes dataset. The training subset is contaminated with extra generating factors (20% of the data is considered as outliers). The framework of [72] Lasso and Random Forest regressor [72] is used to evaluate the learned representation. For disentanglement and completeness higher score is better, for informativeness, lower is better.

Dataset	$h_i dim$	Algorithm	Lasso			Random Forest		
			Disent.	Comple.	Inform.	Disent.	Comple.	Inform.
DSprites	2	β-VAE ($\beta = 3$)	0.19	0.16	6.42	0.13	0.32	**1.39**
		Gen-RKM	0.07	0.07	**5.82**	0.25	0.27	5.91
		Rob Gen-RKM	**0.21**	**0.21**	9.13	**0.36**	**0.38**	5.95
3DShapes	3	β-VAE ($\beta = 3$)	0.24	0.28	**2.72**	0.12	0.13	2.15
		Gen-RKM	0.14	0.14	3.03	0.15	0.15	1.09
		Rob Gen-RKM	**0.47**	**0.49**	3.13	**0.44**	**0.45**	**1.02**

The training data now consist out of 3 'true' generating factors (floor, wall, and object hue) which appear in the majority of the data, and 3 'noisy' generating factors (object, scale, and orientation) which only occur in a small fraction. To illustrate the effect of the weighting scheme, Figure 6.1.4 visualizes the latent space of the (robust) Gen-RKM model. The classical Gen-RKM encodes the outliers in the representation, which results in a distorted Gaussian distribution of the latent variables. This is not the case for the robust Gen-RKM, where the outliers are down-weighted. An illustration of latent traversals along the 3 latent dimensions using the robust Gen-RKM model is given in Figure 6.1.5, where the robust model is capable of disentangling the 3 'clean' generating factors.

To quantify the performance of disentanglement, we use the proposed framework[2] of [72], which consists of 3 metrics: disentanglement, completeness, and informativeness. The framework could be used when the *ground-truth latent structure is available*, which is the case for 3Dshapes and DSprites dataset. The results are shown in Table 6.1.2, where the robust method outperforms the Gen-RKM. The above experiment is repeated on the DSprites dataset. The 'clean' training subset consists of ellipse-shaped data points with minimal scale and 0° angle at different x and y positions. Afterwards, the training data is contaminated with a random sample of different objects at larger scales, different angles at different x and y positions. The training data now consist out of 2 'true' generating factors (x and y positions) which appear in the majority of the data and 3 'noisy' generating factors (orientation, scale, and shape) which only occur in a small fraction. In addition to RKM, the results of β-VAE are shown in Table 6.1.2.

[2]Code and dataset available at `https://github.com/cianeastwood/qedr`

Datasets and model architectures

Table 6.1.3 shows the details on training settings used in this section. The PyTorch library in Python was used with an 8GB NVIDIA QUADRO P4000 GPU.

Table 6.1.3: Model architectures. All convolutions and transposed-convolutions are with stride 2 and padding 1. Unless stated otherwise, layers have Parametric-RELU ($\alpha = 0.2$) activation functions, except output layers of the pre-image maps which have sigmoid activation functions. $N_{\text{sub}} \leq N$ is the training subset size, s the latent space dimension and m the minibatch size.

Dataset	**Optimizer** (Adam)	**Architecture**		**Parameters**	
MNIST/F-MNIST/ CIFAR-10/SVHN	1e-4	Feature-map (fm)	Conv 32×4×4; Conv 64×4×4; FC 228 (Linear)	N_{sub}	3000
		Pre-image map	reverse of fm	s	10
		Latent space dim.	10	m	200
Dsprites/3DShapes	1e-4	Feature-map (fm)	Conv 20×4×4; Conv 40×4×4; Conv 80×4×4; FC 228 (Linear)	N_{sub}	1024/1200
		Pre-image map	reverse of fm	s	2
		Latent space dim.	2/3	m	200

6.2 Outlier detection in non-elliptical data by kernel MRCD

6.2.1 Introduction

The minimum covariance determinant (MCD) estimator introduced in [192, 193] is a robust estimator of multivariate location and covariance. It forms the basis of robust versions of multivariate techniques such as discriminant analysis, principal component analysis, factor analysis and multivariate regression, see e.g. [110, 109] for an overview. The basic MCD method is quite intuitive. Given a data matrix of n rows with d columns, the objective is to find $h < n$ observations whose sample covariance matrix has the lowest determinant. The MCD estimate of location is then the average of those h points, whereas the scatter estimate is a multiple of their covariance matrix. The MCD has good robustness properties. It has a high breakdown value, that is, it can withstand a substantial number of outliers. The effect of a small number of potentially far outliers is measured by its influence function, which is bounded [50].

Computing the MCD was difficult at first but became faster with the algorithm of [199] and the deterministic algorithm DetMCD [112]. An algorithm for n in the millions was recently constructed [53]. But all algorithms for the original MCD require that the dimension d be lower than h in order to obtain an invertible covariance matrix. In fact, it is recommended that $n > 5d$ in practice [199]. This restriction implies that the original MCD cannot be applied to datasets with more variables than cases, that are commonly found in spectroscopy and areas where sample acquisition is difficult or costly, e.g. in the field of omics data.

A solution to this problem was recently proposed in [24], which introduced the minimum regularized covariance determinant (MRCD) estimator. The scatter matrix of a subset of h observations is now a convex combination of its sample covariance matrix and a target matrix. This makes it possible to use the MRCD estimator when the dimension exceeds the subset size. But the computational complexity of MRCD still contains a term $O(d^3)$ from the covariance matrix inversion, which limits its use for high-dimensional data. Another restriction is the assumption that the non-outlying observations roughly follow an elliptical distribution.

To address both issues we propose a generalization of the MRCD which is defined in a kernel induced feature space $\mathcal{F}$, where the proposed estimator exploits the kernel trick: the $d \times d$ covariance matrix is not calculated explicitly but replaced by the calculation of a $n \times n$ centered kernel matrix, resulting in a computational speed-up in case $n \ll d$. Similar ideas can be found in the literature, see e.g. [64, 65] which kernelized the minimum volume ellipsoid [192, 193]. The results of the KMRCD algorithm with the linear kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}$ and radial basis function (RBF) kernel $k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/(2\sigma^2)}$ are shown in Figure 6.2.1. This example will be described in detail in Section 6.2.6.

The section is organized as follows. Section 6.2.2 describes the MCD and MRCD estimators. Section 6.2.3 proposes the kernel MRCD method. Section 6.2.4 describes the kernel-based initial estimators used as well as a kernelized refinement procedure, and proves that the optimization in feature space is equivalent to an optimization in terms of kernel matrices. The simulation study in Section 6.2.5 confirms the robustness of the method as well as the improved computation speed when using a linear kernel. Section 6.2.6 illustrates KMRCD on three datasets, and

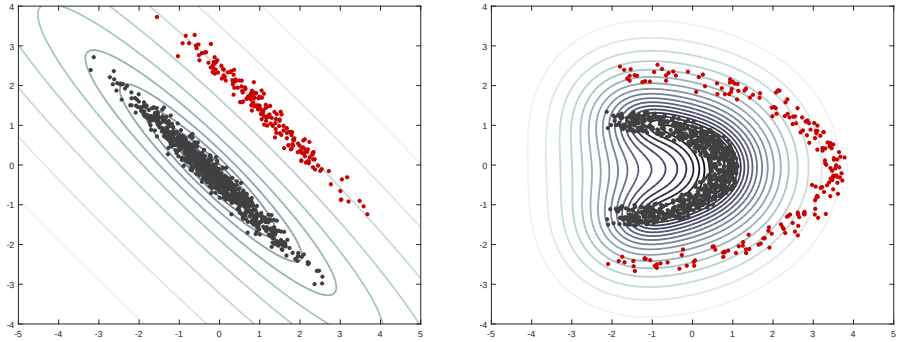


Figure 6.2.1: Illustration of kernel MRCD on two datasets of which the non-outlying part is elliptical (left) and non-elliptical (right). Both datasets contain 20% of outlying observations. The generated regular observations are shown in black and the outliers in red. In the panel on the left, a linear kernel was used, and in the panel on the right a nonlinear kernel. The curves on the left are contours of the robust Mahalanobis distance in the original bivariate space. The contours on the right are based on the robust distance in the kernel-induced feature space.

6.2.2 The MCD and MRCD methods

The Minimum Covariance Determinant estimator

Assume that we have a d-variate dataset X containing n data points, where the i-th observation $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^\top$ is a d-dimensional column vector. We do not know in advance which of these points are outliers, and they can be located anywhere. The objective of the MCD method is to find a set H containing the indices of $|H| = h$ points whose sample covariance matrix has the lowest possible determinant. The user may specify any value of h with $n/2 \leqslant h < n$. The remaining $n - h$ observations could potentially be outliers. For each h-subset H the location estimate $\boldsymbol{c}^H$ is the average of these h points:

$$\boldsymbol{c}^H = \frac{1}{h} \sum_{i \in H} \boldsymbol{x}_i$$

whereas the scatter estimate is a multiple of their covariance matrix, namely

$$c_\alpha \hat{\Sigma}^H = \frac{c_\alpha}{h-1} \sum_{i \in H} (\boldsymbol{x}_i - \boldsymbol{c}^H)(\boldsymbol{x}_i - \boldsymbol{c}^H)^\top$$

where c_α is a consistency factor [50] that depends on the ratio $\alpha = h/n$. The MCD aims to minimize the determinant of $\hat{\Sigma}^H$ among all $H \in \mathcal{H}$, where the

latter denotes the collection of all possible sets H with $|H| = h$:

$$\hat{\Sigma}_{\text{MCD}} = \underset{H \in \mathcal{H}}{\text{argmin}} \ \det(\hat{\Sigma}^H) \ . \tag{6.2.1}$$

Computing the exact MCD has combinatorial complexity, so it is infeasible for all but tiny datasets. However, the approximate algorithm FastMCD constructed in [199] is feasible. FastMCD uses so-called concentration steps (C-steps) to minimize (6.2.1). Starting from any given $\hat{\Sigma}^H$, the C-step constructs a more concentrated approximation by calculating the Mahalanobis distance of every observation based on the location and scatter of the current subset H:

$$\text{MD}(\boldsymbol{x}, \boldsymbol{c}^H, \hat{\Sigma}^H) = \sqrt{(\boldsymbol{x} - \boldsymbol{c}^H)^\top (\hat{\Sigma}^H)^{-1} (\boldsymbol{x} - \boldsymbol{c}^H)} \ .$$

These distances are sorted and the h observations with the lowest distance $\text{MD}(\boldsymbol{x}_i, \boldsymbol{c}^H, \hat{\Sigma}^H)$ form the new h-subset, which is guaranteed to have an equal or lower determinant [199]. The C-step can be iterated until convergence.

The minimum regularized covariance determinant estimator

The minimum regularized covariance determinant estimator (MRCD) is a generalization of the MCD estimator to high dimensional data [24]. The MRCD subset H is defined by minimizing the determinant of the regularized covariance matrix $\hat{\Sigma}^H_{\text{reg}}$:

$$\hat{\Sigma}_{\text{MRCD}} = \underset{H \in \mathcal{H}}{\text{argmin}} \left(\det(\hat{\Sigma}^H_{\text{reg}}) \right) ,$$

where the regularized covariance matrix is given by

$$\hat{\Sigma}^H_{\text{reg}} = \rho T + (1 - \rho) c_\alpha \hat{\Sigma}^H$$

with $0 < \rho < 1$ and T a predetermined and well-conditioned symmetric and positive definite target matrix. The determination of ρ is done in a data-driven way such that $\hat{\Sigma}_{\text{MRCD}}$ has a condition number at most κ, for which [24] proposes $\kappa = 50$. The MRCD algorithm starts from six robust, well-conditioned initial estimates of location and scatter, taken from the DetMCD algorithm [112]. Each initial estimate is followed by concentration steps, and at the end the subset H with the lowest determinant is kept. Note that approximate algorithms like FastMCD and MRCD are much faster than exhaustive enumeration, but one can no longer formally prove a breakdown value. Fortunately, simulations confirm the high robustness of these methods. Also, note that such approximate algorithms are guaranteed to converge, because they iterate C-steps starting from a finite number of initial fits. The algorithm may converge to a local minimum of the objective rather than its global minimum, but simulations have confirmed the accuracy of the result.

6.2.3 The kernel MRCD estimator

We now turn our attention to kernel transformations [209], formally defined as follows.

Definition 6.2.1. *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel on $\mathcal{X}$ iff there exists a real Hilbert space $\mathcal{F}$ and a map $\phi : \mathcal{X} \to \mathcal{F}$ such that for all $\boldsymbol{x}$, $\boldsymbol{y}$ in $\mathcal{X}$:*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle,$$

where ϕ is called a feature map and $\mathcal{F}$ is called a feature space.

We restrict ourselves to positive semidefinite (PSD) kernels. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called PSD iff $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geqslant 0$ for any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in $\mathcal{X}$ and any $c_1, \ldots, c_n$ in $\mathbb{R}$. Given an $n \times d$ dataset X, its kernel matrix is defined as $K = \Phi\Phi^\top$ with $\Phi = [\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_n)]^\top$. The use of kernels makes it possible to operate in a high-dimensional, implicit feature space without computing the coordinates of the data in that space, but rather by replacing inner products by kernel matrix entries. A well-known example is given by kernel PCA [208], where linear PCA is performed in a kernel-induced feature space $\mathcal{F}$ instead of the original space $\mathcal{X}$. Working with kernel functions has the the advantage that non-linear kernels enable the construction of non-linear models. Note that the size of the kernel matrix is $n \times n$, whereas the covariance matrix is $d \times d$. The latter is an advantage when dealing with datasets for which $n \ll d$, for then the memory and computational requirements are considerably lower.

Given an $n \times d$ dataset $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ we thus get its image $\{\phi(\boldsymbol{x}_1), \ldots \phi(\boldsymbol{x}_n)\}$ in feature space, where it has the average

$$\boldsymbol{c}_{\mathcal{F}} = \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_i) \; .$$

Note that the dimension of the feature space $\mathcal{F}$ may be infinite. However, we will restrict ourselves to the subspace $\tilde{\mathcal{F}}$ spanned by $\{\phi(\boldsymbol{x}_1) - \boldsymbol{c}_{\mathcal{F}}, \ldots, \phi(\boldsymbol{x}_n) - \boldsymbol{c}_{\mathcal{F}}\}$ so that $m := \dim(\tilde{\mathcal{F}}) \leqslant n - 1$. In this subspace the points $\phi(\boldsymbol{x}_i) - \boldsymbol{c}_{\mathcal{F}}$ thus have at most $n - 1$ coordinates. The covariance matrix in the feature space given by

$$\hat{\Sigma}_{\mathcal{F}} = \frac{1}{n-1} \sum_{i=1}^{n} (\phi(\boldsymbol{x}_i) - \boldsymbol{c}_{\mathcal{F}})(\phi(\boldsymbol{x}_i) - \boldsymbol{c}_{\mathcal{F}})^\top$$

is thus a matrix of size at most $(n-1) \times (n-1)$. Note that the covariance matrix is centered but the original kernel matrix is not. Therefore we construct

the centered kernel matrix $\tilde{K}$ by

$$\begin{aligned}\tilde{K}_{ij} &= \Big(\phi(\boldsymbol{x}_i) - \frac{1}{n}\sum_{\ell=1}^{n}\phi(\boldsymbol{x}_\ell)\Big)^\top \Big(\phi(\boldsymbol{x}_j) - \frac{1}{n}\sum_{\ell'=1}^{n}\phi(\boldsymbol{x}_{\ell'})\Big) \\ &= K_{ij} - \frac{1}{n}\sum_{\ell=1}^{n}K_{\ell j} - \frac{1}{n}\sum_{\ell'=1}^{n}K_{i\ell'} + \frac{1}{n^2}\sum_{\ell=1}^{n}\sum_{\ell'=1}^{n}K_{\ell\ell'} \\ &= \Big(K - 1_{nn}K - K1_{nn} + 1_{nn}K1_{nn}\Big)_{ij} \qquad (6.2.2)\end{aligned}$$

where 1_{nn} is the $n \times n$ matrix with all entries set to $1/n$. Note that the centered kernel matrix is equal to $\tilde{K} = \tilde{\Phi}\tilde{\Phi}^\top$ with $\tilde{\Phi} = [\phi(\boldsymbol{x}_1) - \boldsymbol{c}_{\mathcal{F}}, \ldots, \phi(\boldsymbol{x}_n) - \boldsymbol{c}_{\mathcal{F}}]^\top$ and is PSD by construction. The following result is due to [208].

Theorem 6.2.1. *Given an $n \times p$ dataset X, the sorted eigenvalues of the covariance matrix $\hat{\Sigma}_{\mathcal{F}}$ and those of the centered kernel matrix $\tilde{K}$ satisfy*

$$\lambda_j^{\hat{\Sigma}_{\mathcal{F}}} = \frac{\lambda_j^{\tilde{K}}}{n-1}$$

for all $j = 1, \ldots, m$ where $m = \operatorname{rank}(\hat{\Sigma}_{\mathcal{F}})$.

Proof of Theorem 6.2.1. The eigendecomposition of the centered kernel matrix $\tilde{K}$ is

$$\tilde{K} = \tilde{\Phi}\tilde{\Phi}^\top = V\Lambda V^\top$$

where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_1 \geqslant \ldots \geqslant \lambda_n$. The eigenvalue λ_j and eigenvector $\boldsymbol{v}_j$ satisfy

$$\tilde{\Phi}\tilde{\Phi}^\top \boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j$$

for all $j = 1, \ldots, m$. Multiplying both sides by $\tilde{\Phi}^\top/(n-1)$ gives

$$\left(\frac{1}{n-1}\tilde{\Phi}^\top\tilde{\Phi}\right)(\tilde{\Phi}^\top \boldsymbol{v}_j) = \frac{\lambda_j}{n-1}(\tilde{\Phi}^\top \boldsymbol{v}_j).$$

Combining the above equations results in

$$\hat{\Sigma}_{\mathcal{F}}\boldsymbol{v}_j^{\hat{\Sigma}_{\mathcal{F}}} = \frac{\lambda_j}{n-1}\boldsymbol{v}_j^{\hat{\Sigma}_{\mathcal{F}}}$$

for all $j = 1, \ldots, m$ where $\boldsymbol{v}_j^{\hat{\Sigma}_{\mathcal{F}}} = (\tilde{\Phi}^\top \boldsymbol{v}_j)$ is the j-th eigenvector of $\hat{\Sigma}_{\mathcal{F}}$. The remaining eigenvalues of the covariance matrix, if any, are equal to zero. □

The above result can be related to a representer theorem for kernel PCA [10]. It shows that the nonzero eigenvalues of the covariance matrix are proportional to the nonzero eigenvalues of the centered kernel matrix, thus proving that $\hat{\Sigma}_{\mathcal{F}}$ and $\tilde{K}$ have the same rank.

What would a kernelized MCD estimator look like? It would have to be equivalent to applying the original MCD in the feature space, so that in case of the linear kernel the original MCD is obtained. The MCD estimate for location in the subspace $\tilde{\mathcal{F}}$ is

$$\boldsymbol{c}_{\mathcal{F}}^{H} = \frac{1}{h}\sum_{i\in H}\phi(\boldsymbol{x}_i)$$

whereas the covariance matrix now equals

$$\hat{\Sigma}_{\mathcal{F}}^{H} = \frac{1}{h-1}\sum_{i\in H}(\phi(\boldsymbol{x}_i)-\boldsymbol{c}_{\mathcal{F}}^{H})(\phi(\boldsymbol{x}_i)-\boldsymbol{c}_{\mathcal{F}}^{H})^{\top}.$$

Likewise, the robust distance becomes

$$\mathrm{MD}(\phi(\boldsymbol{x}),\boldsymbol{c}_{\mathcal{F}}^{H},\hat{\Sigma}_{\mathcal{F}}^{H}) = (\phi(\boldsymbol{x})-\boldsymbol{c}_{\mathcal{F}}^{H})^{\top}(\hat{\Sigma}_{\mathcal{F}}^{H})^{-1}(\phi(\boldsymbol{x})-\boldsymbol{c}_{\mathcal{F}}^{H})\ .$$

In these formulas the mapping function ϕ may not be known, but that is not necessary since we can apply the kernel trick. More importantly, the covariance matrix may not be invertible as the $\phi(\boldsymbol{x}_i)-\boldsymbol{c}_{\mathcal{F}}^{H}$ lie in a possibly high-dimensional space $\tilde{\mathcal{F}}$. We, therefore, propose to apply MRCD in $\tilde{\mathcal{F}}$ in order to make the covariance matrix invertible. Let $\tilde{\Phi}_H$ be the row-wise stacked matrix

$$\tilde{\Phi}_H = [\phi(\boldsymbol{x}_{i(1)})-\boldsymbol{c}_{\mathcal{F}}^{H},\ldots,\phi(\boldsymbol{x}_{i(h)})-\boldsymbol{c}_{\mathcal{F}}^{H}]^{\top}$$

where $i(1),\ldots,i(h)$ are the indices in H. For any $0<\rho<1$ the regularized covariance matrix is defined as

$$\hat{\Sigma}_{\mathrm{reg}}^{H} = (1-\rho)\hat{\Sigma}_{\mathcal{F}}^{H}+\rho\mathbb{I}_m = \frac{1-\rho}{h-1}\tilde{\Phi}_H^{\top}\tilde{\Phi}_H+\rho\mathbb{I}_m$$

where $\mathbb{I}_m$ is the identity matrix in $\tilde{\mathcal{F}}$. The KMRCD method is then defined as

$$\hat{\Sigma}_{\mathrm{KMRCD}} = \underset{H\in\mathcal{H}}{\operatorname{argmin}}\det(\hat{\Sigma}_{\mathrm{reg}}^{H}) \tag{6.2.3}$$

where $\mathcal{H}$ is the collection of subsets H of $\{1,\ldots,n\}$ such that $|H|=h$ and $\hat{\Sigma}^{H}$ is of maximal rank, i.e. $\mathrm{rank}(\hat{\Sigma}^{H}) = \dim(\mathrm{span}(\phi(\boldsymbol{x}_{i(1)})-\boldsymbol{c}_{\mathcal{F}}^{H},\ldots,\phi(\boldsymbol{x}_{i(h)})-\boldsymbol{c}_{\mathcal{F}}^{H})) = q$ with $q:=\min(m,h-1)$. We can equivalently say that the h-subset H is in general position. The corresponding regularized kernel matrix is

$$\tilde{K}_{\mathrm{reg}}^{H} = (1-\rho)\tilde{K}^{H}+(h-1)\rho\mathbb{I}_h \tag{6.2.4}$$

where $\tilde{K}^H = \tilde{\Phi}_H \tilde{\Phi}_H^{\mathrm{T}}$ denotes the centered kernel matrix of h rows, that is, (6.2.2) with n replaced by h. The MRCD method in feature space $\tilde{\mathcal{F}}$ minimizes the determinant in (6.2.3) in $\tilde{\mathcal{F}}$. But we would like to carry out an optimization on kernel matrices instead. The following theorem shows that this is possible.

Theorem 6.2.2. *Minimizing* $\det(\hat{\Sigma}^H_{\mathrm{reg}})$ *over all subsets* H *in* $\mathcal{H}$ *is equivalent to minimizing* $\det(\tilde{K}^H_{\mathrm{reg}})$ *over all* h*-subsets* H *with* $\operatorname{rank}(\tilde{K}^H) = q$.

Proof of Theorem 6.2.2. From Theorem 6.2.1 it follows that the nonzero eigenvalues of $\hat{\Sigma}^H_{\mathcal{F}}$ and $\tilde{K}^H$ are related by $\lambda_j^{\hat{\Sigma}^H_{\mathcal{F}}} = \frac{1}{h-1}\lambda_j^{\tilde{K}^H}$. If H belongs to $\mathcal{H}$, $\hat{\Sigma}^H_{\mathcal{F}}$ has exactly q nonzero eigenvalues so $\tilde{K}^H$ also has rank q, and vice versa. The remaining $m-q$ eigenvalues of $\hat{\Sigma}^H$ are zero, as well as the remaining $h-q$ eigenvalues of $\tilde{K}^H$. Now consider the regularized matrices

$$\hat{\Sigma}^H_{\mathrm{reg}} = (1-\rho)\hat{\Sigma}^H_{\mathcal{F}} + \rho \mathbb{I}_m$$

and

$$\tilde{K}^H_{\mathrm{reg}} = (1-\rho)\tilde{K}^H + (h-1)\rho \mathbb{I}_h \ .$$

Computing the determinant of both matrices as a product of their eigenvalues yields:

$$\det(\hat{\Sigma}^H_{\mathrm{reg}}) = \rho^{m-q} \prod_{j=1}^{q} ((1-\rho)\lambda_j^{\hat{\Sigma}^H_{\mathcal{F}}} + \rho)$$

and

$$\begin{aligned}
\det(\tilde{K}^H_{\mathrm{reg}}) &= \rho^{h-q} \prod_{j=1}^{q} ((1-\rho)\lambda_j^{\tilde{K}^H} + (h-1)\rho) \\
&= \rho^{h-q} \prod_{j=1}^{q} (h-1)((1-\rho)\lambda_j^{\hat{\Sigma}^H_{\mathcal{F}}} + \rho) \\
&= \frac{\rho^{h-q}}{\rho^{m-q}} (h-1)^q \det(\hat{\Sigma}^H_{\mathrm{reg}}) \ .
\end{aligned}$$

Therefore $\det(\tilde{K}^H_{\mathrm{reg}}) = \rho^{h-m}(h-1)^q \det(\hat{\Sigma}^H_{\mathrm{reg}})$ in which the proportionality factor is constant, so the optimizations are equivalent. □

Following [93] we can also express the robust Mahalanobis distance in terms of the regularized kernel matrix, by

$$\begin{aligned}\mathrm{MD}(\phi(\boldsymbol{x}), \boldsymbol{c}_{\mathcal{F}}^{H}, \hat{\Sigma}_{\mathrm{reg}}^{H}) &= \sqrt{(\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{F}}^{H})^{\top}(\hat{\Sigma}_{\mathrm{reg}}^{H})^{-1}(\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{F}}^{H})} \\ &= \sqrt{\frac{1}{\rho}\left(\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) - (1-\rho)\tilde{k}(H, x)^{\top}(\tilde{K}_{\mathrm{reg}}^{H})^{-1}\tilde{k}(H, \boldsymbol{x})\right)}\end{aligned} \tag{6.2.5}$$

where $\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) = (\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{F}}^{H})^{\top}(\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{F}}^{H})$ is a special case of the formula $\tilde{k}(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}, \boldsymbol{y}) - \sum_{i \in H} k(\boldsymbol{x}_i, \boldsymbol{x}) - \sum_{i \in H} k(\boldsymbol{x}_i, \boldsymbol{y}) - \sum_{i \in H}\sum_{j \in H} k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $\boldsymbol{x} = \boldsymbol{y}$. The notation $\tilde{k}(H, x)$ stands for the column vector $\tilde{\Phi}_H(\phi(\boldsymbol{x}) - \boldsymbol{c}_{\mathcal{F}}^{H}) = [\tilde{k}(\boldsymbol{x}_{i(1)}, \boldsymbol{x}), \ldots, \tilde{k}(\boldsymbol{x}_{i(h)}, \boldsymbol{x})]^{\top}$ in which $i(1), \ldots, i(h)$ are the members of H. This allows us to calculate the Mahalanobis distance in feature space from the kernel matrix, and consequently to perform the C-step procedure on it. Note that (6.2.5) requires inverting the matrix $\tilde{K}_{\mathrm{reg}}^{H}$ instead of the matrix $\hat{\Sigma}_{\mathrm{reg}}^{H}$.

The C-step theorem of the MRCD in [24] shows that when you select a new h-subset as those i for which the Mahalanobis distance relative to the old h-subset is smallest, the regularized covariance determinant of the new h-subset is lower than or equal to that of the old one. In other words, C-steps lower the objective function of MRCD. Using Theorem 6.2.2, this C-step theorem thus also extends to the kernel MRCD estimator.

6.2.4 The kernel MRCD algorithm

This section introduces the elements of the kernel MRCD algorithm. If the original data comes in the form of an $n \times d$ dataset X, we start by robustly standardizing it. For this, we first compute the univariate reweighted MCD estimator of [196] with coverage $h = [n/2]+1$ to obtain estimates of the location and scatter of each variable, which are then used to transform X to z-scores. The kernel matrix K is then computed from these z-scores. Note, however, that the data can come in the form of a kernel matrix that was not derived from data points with coordinates. For instance, a so-called string kernel can compute similarities between texts, such as emails, without any variables or measurements. Such a kernel basically compares the occurrence of strings of consecutive letters in each text. Since the KMRCD method does all its computations on the kernel matrix, it can also be applied to such data.

Initial estimates

The MRCD estimator needs initial h-subsets to start C-steps from. In the original FastMCD algorithm of [199] the initial h-subsets were obtained by drawing random $(d+1)$-subsets out of the n data points. For each its empirical mean and covariance matrix was computed as well as the resulting Mahalanobis distances of all points, after which the subset with the h smallest distances were obtained. However, this procedure would not be possible in situations where $d > n$ because Mahalanobis distances require the covariance matrix to be invertible. The MRCD method instead starts from a small number of other initial estimators, inherited from the DetMCD algorithm in [112].

For the initial h-subsets in KMRCD we need methods that can be kernelized. We propose to use four such initial estimators, the combination of which has a good chance of being robust against different contamination types. Since initial estimators can be inaccurate, a kernelized refinement step will be applied to each. We will describe these methods in turn.

The first initial method is based on the concept of spatial median. For data with coordinates, the spatial median is defined as the point $\boldsymbol{m}$ that has the lowest total Euclidean distance $\sum_i ||\boldsymbol{x}_i - \boldsymbol{m}||$ to the data points. This notion also makes sense in the kernel context, since Euclidean distances in the feature space can be written in terms of the inner products that make up the kernel matrix. The spatial median in coordinate space is often computed by the Weiszfeld algorithm and its extensions, see e.g. [237]. A kernel algorithm for the spatial median was provided in [54]. It writes the spatial median $m_{\mathcal{F}}$ in feature space as a convex combination of the $\phi(\boldsymbol{x}_i)$:

$$\boldsymbol{m}_{\mathcal{F}} = \sum_{i=1}^{n} \gamma_i \phi(\boldsymbol{x}_i)$$

in which the coefficients $\gamma_1, \ldots, \gamma_n$ are unknown. The Euclidean distance of each observation to $\boldsymbol{m}_{\mathcal{F}}$ is computed as the square root of

$$\begin{aligned} ||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_{\mathcal{F}}||^2 &= ||\phi(\boldsymbol{x}_i) - \sum_{j=1}^{n} \gamma_j \phi(\boldsymbol{x}_j)||^2 \\ &= ||\phi(\boldsymbol{x}_i)||^2 + ||\sum_{j=1}^{n} \gamma_j \phi(\boldsymbol{x}_j)||^2 - 2\langle \phi(\boldsymbol{x}_i), \sum_{j=1}^{n} \gamma_j \phi(\boldsymbol{x}_j) \rangle \\ &= k(\boldsymbol{x}_i, \boldsymbol{x}_i) + \sum_{j=1}^{n} \sum_{\ell=1}^{n} \gamma_j \gamma_\ell k(\boldsymbol{x}_j, \boldsymbol{x}_\ell) - 2 \sum_{j=1}^{n} \gamma_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{6.2.6}$$

and the coefficients $\gamma_1, \ldots, \gamma_n$ that minimize $\sum_i ||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_{\mathcal{F}}||$ are obtained by an iterative procedure described in Appendix G.1. The first initial h-subset H is then given by the objects with the h lowest values of (6.2.6). Alternatively, H is described by a weight vector $\boldsymbol{w} = (w_1, \ldots, w_n)$ of length n, where

$$w_i := \begin{cases} 1 & \text{if } i \in H \\ 0 & \text{otherwise.} \end{cases} \tag{6.2.7}$$

The initial location estimate $\boldsymbol{c}_{\mathcal{F}}$ in feature space is then the weighted mean

$$\boldsymbol{c}_{\mathcal{F}} = \frac{\sum_{i=1}^n w_i \phi(\boldsymbol{x}_i)}{\sum_{i=1}^n w_i} \,. \tag{6.2.8}$$

The initial covariance estimate $\hat{\Sigma}_{\mathcal{F}}$ is the weighted covariance matrix

$$\hat{\Sigma}_{\mathcal{F}} = \frac{1}{\sum_{i=1}^n u_i} \tilde{\Phi}^\top \mathrm{diag}(u_1, \ldots, u_n)\, \tilde{\Phi} \tag{6.2.9}$$

given by covariance weights $(u_1, \ldots, u_n)$ that in general may differ from the location weights $(w_1, \ldots, w_n)$. But for the spatial median initial estimator one simply takes $u_i := w_i$ for all i.

The second initial estimator is based on the Stahel-Donoho outlyingness (SDO) of [219, 66]. In a space with coordinates it involves projecting the data points on many unit length vectors (directions). We compute the kernelized SDO [56] of all observations and determine an h-subset as the indices of the h points with the lowest outlyingness. This is then converted to weights w_i as in (6.2.7), and we put $u_i := w_i$ again. The entire procedure is given in Appendix G.1.

The third initial h-subset is based on spatial ranks [55]. The spatial rank of $\phi(\boldsymbol{x}_i)$ with respect to the other feature vectors is defined as:

$$\begin{aligned} R_i &= \frac{1}{n} \left\| \sum_{j \neq i} \frac{\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)}{\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|} \right\| \\ &= \frac{1}{n} \left[\left(\sum_{j \neq i} \frac{\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)}{\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|} \right)^\top \left(\sum_{\ell \neq i} \frac{\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_\ell)}{\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_\ell)\|} \right) \right]^{\frac{1}{2}} \\ &= \frac{1}{n} \left[\sum_{j \neq i} \sum_{\ell \neq i} \frac{k(\boldsymbol{x}_i, \boldsymbol{x}_i) - k(\boldsymbol{x}_i, \boldsymbol{x}_j) - k(\boldsymbol{x}_i, \boldsymbol{x}_\ell) + k(\boldsymbol{x}_j, \boldsymbol{x}_\ell)}{\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha(\boldsymbol{x}_i, \boldsymbol{x}_\ell)} \right]^{\frac{1}{2}} \end{aligned} \tag{6.2.10}$$

where $\alpha(\boldsymbol{x}_i, x_j) = [k(\boldsymbol{x}_i, \boldsymbol{x}_i) + k(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j)]^{\frac{1}{2}}$. If R_i is large, this indicates that $\phi(\boldsymbol{x}_i)$ lies further away from the bulk of the data than most other feature vectors. In this sense, the values R_i represent a different measure of the outlyingness of $\phi(\boldsymbol{x}_i)$ in the feature space. We then consider the h lowest spatial ranks, yielding the location weights w_i by (6.2.7), and put $u_i := w_i$. The complete procedure is given in Appendix G.1. Note that this algorithm is closely related to the depth computation in [44] which appeared in the same year as [55].

The last initial estimator is a generalization of the spatial sign covariance matrix [238] (SSCM) to the feature space $\mathcal{F}$. For data with coordinates, one first computes the spatial median $\boldsymbol{m}$ described above. The SSCM then carries out a radial transform which moves all data points to a sphere around $\boldsymbol{m}$, followed by computing the classical product moment of the transformed data:

$$\hat{\Sigma}^{\text{SSCM}} = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(\boldsymbol{x}_i - \boldsymbol{m})}{||\boldsymbol{x}_i - \boldsymbol{m}||} \frac{(\boldsymbol{x}_i - \boldsymbol{m})^\top}{||\boldsymbol{x}_i - \boldsymbol{m}||} .$$

The kernel spatial sign covariance matrix [54] is defined in the same way, by replacing $\boldsymbol{x}_i$ by $\phi(\boldsymbol{x}_i)$ and $\boldsymbol{m}$ by $\boldsymbol{m}_{\mathcal{F}} = \sum_{i=1}^{n} \gamma_i \phi(\boldsymbol{x}_i)$. We now have two sets of weights. For location we use the weights $w_i = \gamma_i$ of the spatial median and apply (6.2.8). But for the covariance matrix we compute the weights $u_i = 1/||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_{\mathcal{F}}||$ with the denominator given by (6.2.6). Next, we apply (6.2.9) with these u_i . The entire kernel SSCM procedure is given in Appendix G.1. Note that kernel SSCM uses continuous weights instead of zero-one weights.

The refinement step

It happens that the eigenvalues of initial covariance estimators are inaccurate. In [152] this was addressed by re-estimating the eigenvalues, and [112] carried out this refinement step for all initial estimates used in that paper. In order to employ a refinement step in KMRCD we need to be able to kernelize it. We will derive the equations for the general case of a location estimator given by a weighted sum (6.2.8) and a scatter matrix estimate given by a weighted covariance matrix (6.2.9) so it can be applied to all four initial estimates. We proceed in four steps.

1. The first step consists of projecting the uncentered data on the eigenvectors $V_{\mathcal{F}}$ of the initial scatter estimate $\hat{\Sigma}_{\mathcal{F}}$:

$$B = \Phi V_{\mathcal{F}} = \Phi \tilde{\Phi}^\top D^{\frac{1}{2}} V = (K - K\boldsymbol{w}\mathbf{1}_n^\top) D^{\frac{1}{2}} V, \tag{6.2.11}$$

where $D = \text{diag}(u_1, \ldots, u_n)/(\sum_{i=1}^n u_i)$, $1_n = [1, \ldots, 1]^\top$, and $V_{\mathcal{F}} = \tilde{\Phi}^\top D^{\frac{1}{2}} V$ with V the normalized eigenvectors of the weighted centered kernel matrix $\hat{K} = (D^{\frac{1}{2}}\tilde{\Phi})(D^{\frac{1}{2}}\tilde{\Phi})^\top = D^{\frac{1}{2}}\tilde{K}D^{\frac{1}{2}}$.

2. Next, the covariance matrix is re-estimated by

$$\Sigma^*_{\mathcal{F}} = V_{\mathcal{F}} L V_{\mathcal{F}}^\top = \tilde{\Phi}^\top D^{\frac{1}{2}} V L V^\top D^{\frac{1}{2}} \tilde{\Phi} \,,$$

where $L = \text{diag}(Q_n^2\,(B_{.1}), \ldots, Q_n^2\,(B_{.n}))$ in which Q_n is the scale estimator of Rousseeuw and Croux [194] and $B_{.j}$ is the j-th column of B.

3. The center is also re-estimated, by

$$\boldsymbol{c}^*_{\mathcal{F}} = (\Sigma^*_{\mathcal{F}})^{\frac{1}{2}}\, \text{median}(\Phi(\Sigma^*_{\mathcal{F}})^{-\frac{1}{2}})$$

where median stands for the spatial median. This corresponds to using a modified feature map $\phi^*(\boldsymbol{x}) = \phi(\boldsymbol{x})(\Sigma^*_{\mathcal{F}})^{-\frac{1}{2}}$ for the spatial median or running Algorithm G.1.3 with the modified kernel matrix

$$\begin{aligned} K^* &= \Phi\tilde{\Phi}^\top D^{\frac{1}{2}} V L^{-1} V^\top D^{\frac{1}{2}} \tilde{\Phi}\Phi^\top \\ &= (K - K\boldsymbol{w}\mathbf{1}_n^\top) D^{\frac{1}{2}} V L^{-1} D^{\frac{1}{2}} (K - K\boldsymbol{w}\mathbf{1}_n^\top)^\top. \end{aligned} \tag{6.2.12}$$

Transforming the spatial median gives us the desired center:

$$\boldsymbol{c}^*_{\mathcal{F}} = (\Sigma^*_{\mathcal{F}})^{\frac{1}{2}} \sum_{i=1}^n (\Sigma^*_{\mathcal{F}})^{-\frac{1}{2}} \gamma_i^* \phi(\boldsymbol{x}_i) = \sum_{i=1}^n \gamma_i^* \phi(\boldsymbol{x}_i),$$

where γ_i^* are the weights of the spatial median for the modified kernel matrix.

4. The kernel Mahalanobis distance is calculated as

$$\begin{aligned} d^*_{\mathcal{F}}(\boldsymbol{x}) &= (\phi(\boldsymbol{x}) - \boldsymbol{c}^*_{\mathcal{F}})^\top (\Sigma^*_{\mathcal{F}})^{-1} (\phi(\boldsymbol{x}) - \boldsymbol{c}^*_{\mathcal{F}}) \\ &= (\phi(\boldsymbol{x}) - \boldsymbol{c}^*_{\mathcal{F}})^\top \tilde{\Phi}^\top D^{\frac{1}{2}} V L^{-1} V^\top D^{\frac{1}{2}} \tilde{\Phi} (\phi(\boldsymbol{x}) - \boldsymbol{c}^*_{\mathcal{F}}) \\ &= k^*(\boldsymbol{x}, X) D^{\frac{1}{2}} V L^{-1} V^\top D^{\frac{1}{2}} k^*(\boldsymbol{x}, X)^\top \end{aligned} \tag{6.2.13}$$

with

$$\begin{aligned} k^*(\boldsymbol{x}, X) = k(\boldsymbol{x}, X) &- \sum_{i=1}^n w_i k(\boldsymbol{x}, \boldsymbol{x}_i) \mathbf{1}_n^\top \\ &- \sum_{j=1}^n \gamma_j^* k(\boldsymbol{x}_j, X) - \sum_{i=1}^n \sum_{j=1}^n w_i \gamma_j^* k(\boldsymbol{x}_i, \boldsymbol{x}_j) \mathbf{1}_n^\top \end{aligned}$$

where $k(\boldsymbol{x}, X) = [k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_n)]$.

The h points with the smallest $d^*_{\mathcal{F}}(\boldsymbol{x})$ form the refined h-subset. The entire procedure is given in Appendix G.1.

Kernel MRCD algorithm

We now have all the elements to compute the kernel MRCD by Algorithm 6.2.1. Given any PSD kernel matrix and subset size h, the algorithm starts by computing the four initial estimators described in Section 6.2.4. Each initial estimate is then refined according to Section 6.2.4. Next, kernel MRCD computes the regularization parameter ρ. This is done with a kernelized version of the procedure in [24]. For each initial estimate we choose ρ such that the regularized kernel matrix $\tilde{K}^H_{\text{reg}}$ of (6.2.4) is well-conditioned. If we denote by λ the vector containing the eigenvalues of the centered kernel matrix $\tilde{K}^H$, the condition number of $\tilde{K}^H_{\text{reg}}$ is

$$\kappa(\rho) = \frac{(h-1)\rho + (1-\rho)\max(\lambda)}{(h-1)\rho + (1-\rho)\min(\lambda)} \tag{6.2.14}$$

and we choose ρ such that $\kappa(\rho) \leqslant 50$. (The Supplementary Material contains a simulation study supporting this choice.) Finally, kernel C-steps are applied until convergence, where we monitor the objective function of Section 6.2.3.

Algorithm 6.2.1 Kernel MRCD.

Input: kernel matrix K, subset size h.
Compute the weights of the four initial estimates of location and scatter as in Section 6.2.4.
Refine each initial estimate as in Section 6.2.4.
For each resulting subset, determine $\rho^{(i)}$ such that $\kappa(\rho^{(i)}) \leqslant 50$.
Determine the final ρ as in [24]: if $\max_i \rho^{(i)} \leqslant 0.1$ set $\rho = \max_i \rho^{(i)}$, otherwise set $\rho = \max\big(0.1, \text{median}_i(\rho^{(i)})\big)$.
For $H = H^{(1)}, \ldots, H^{(4)}$ perform C-steps as follows:

1. Compute the regularized kernel matrix $\tilde{K}^H_{\text{reg}}$ for the h-subset H from (6.2.4).
2. Calculate the regularized Mahalanobis distance for each observation i by (6.2.5).
3. Redefine H as the h indices i with smallest distance.
4. Compute and store the objective. If not converged, go back to (a).

Select the h-subset with the overall smallest objective function.
Output: the final h-subset and the robust distances.

In the special case where the linear kernel is used, the centered kernel matrix $\tilde{K}^H$ immediately yields the regularized covariance matrix $\hat{\Sigma}^H_{\text{reg}}$ through

$$\hat{\Sigma}^H_{\text{reg}} = \frac{1-\rho}{h-1}(\tilde{X}^H)^\top \tilde{V}\Lambda\tilde{V}^\top \tilde{X}^H + \rho\mathbb{I}_h$$

where $\tilde{X}^H = X^H - \frac{1}{h}\sum_{i\in H} \boldsymbol{x}_i$ is the centered matrix of the observations in H and Λ and $\tilde{V}$ contain the eigenvalues and normalized eigenvectors of $\tilde{K}^H$. (The derivation is given in Appendix G.2.) So instead of applying MRCD to coordinate data we can also run KMRCD with a linear kernel and transform $\tilde{K}^H$ to $\hat{\Sigma}^H_{\text{reg}}$ afterward. This computation is faster when the data has more dimensions than cases.

Anomaly detection by KMRCD

Mahalanobis distances (MD) relative to robust estimates of location and scatter are very useful to flag outliers, because outlying points i tend to have higher MD_i values. The standard way to detect outliers by means of the MCD in low dimensional data is to compare the robust distances to a cutoff that is the square root of a quantile of the chi-squared distribution with degrees of freedom equal to the data dimension [199]. However, in high dimensions the distribution of the squared robust distances is no longer approximately chi-squared, which makes it harder to determine a suitable cutoff value. Faced with a similar problem [197] introduced a different approach, based on the empirical observation that robust distances of the non-outliers in higher dimensional data tend to have a distribution that is roughly similar to a lognormal. They first transform the distances MD_i to $\text{LD}_i = \log(0.1 + \text{MD}_i)$, where the term 0.1 prevents numerical problems should a (near-)zero MD_i occur. The location and spread of the non-outlying LD_i are then estimated by $\hat{\mu}_{\text{MCD}}$ and $\hat{\sigma}_{\text{MCD}}$, the results of applying the univariate MCD to all LD_i using the same h as in the KMCRD method itself. Data point i is then flagged iff

$$\frac{\text{LD}_i - \hat{\boldsymbol{\mu}}_{\text{MCD}}(\text{LD})}{\hat{\sigma}_{\text{MCD}}(\text{LD})} > z(0.995)$$

where $z(0.995)$ is the 0.995 quantile of the standard normal distribution. The cutoff value for the untransformed robust distances is thus

$$c = \exp\left(\hat{\boldsymbol{\mu}}_{\text{MCD}}(\text{LD}) + z(0.995)\hat{\sigma}_{\text{MCD}}(\text{LD})\right) - 0.1\ . \tag{6.2.15}$$

The user may want to try different values of h to be used in both the KMRCD method itself as well as in the $\hat{\boldsymbol{\mu}}_{\text{MCD}}$ and $\hat{\sigma}_{\text{MCD}}$ in (6.2.15). One typically starts with a rather low value of h, say $h = 0.5n$ when the linear kernel is used and

there are up to 10 dimensions, and $h = 0.75n$ in all other situations. This will provide an idea about the number of outliers in the data, after which it is recommended to choose h as high as possible provided $n - h$ exceeds the number of outliers. This will improve the accuracy of the estimates.

Choice of bandwidth

A commonly used kernel function is the radial basis function (RBF) $k(\boldsymbol{x}, \boldsymbol{y}) = e^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/(2\sigma^2)}$ which contains a tuning constant σ that needs to be chosen. When the downstream learning task is classification σ is commonly selected by cross validation, where it is assumed that the data has no outliers or they have already been removed. However, in our unsupervised outlier detection context there is nothing to cross validate. Therefore, we will use the so-called median heuristic [90] given by

$$\sigma^2 = \text{median}\{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \; ; \; 1 \leqslant i < j \leqslant n\} \tag{6.2.16}$$

in which the $\boldsymbol{x}_i$ are the standardized data in the original space. We will use this σ in all our examples.

Illustration on toy examples

We illustrate the proposed KMRCD method on the two toy examples in Figure 6.2.1. Both datasets consist of $n = 1000$ bivariate observations. The elliptical dataset in the left panel was generated from a bivariate Gaussian distribution, plus 20% of outliers. The non-elliptical dataset in the panel on the right is frequently used to demonstrate kernel methods [225]. This dataset also contains 20% of outliers, which are shown in red and form the outer curved shape. We first apply the non-kernel MCD method, which does not transform the data, with $h = \lfloor 0.75n \rfloor$. (Not using a kernel is equivalent to using the linear kernel.) The results are in Figure 6.2.2. In the panel on the left this works well because the MCD method was developed for data of which the majority has a roughly elliptical shape. For the same reason it does not work well on the non-elliptical data in the right hand panel.

We now apply the kernel MRCD method to the same datasets. For the elliptical dataset we use the linear kernel, and for the non-elliptical dataset we use the RBF kernel with tuning constant σ given by formula (6.2.16). This yields Figure 6.2.3. We first focus on the left hand column. The figure shows three stages of the KMRCD runs. At the top, in Figure 6.2.3(a), we see the result for the selected h-subset, after the C-steps have converged. The members of that h-subset are the green points, whereas the points generated as outliers

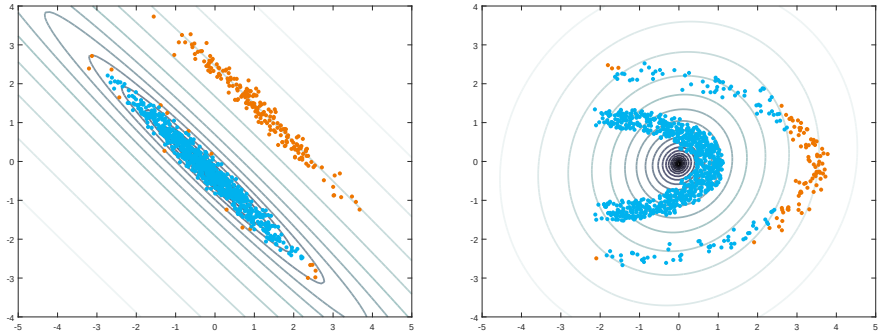
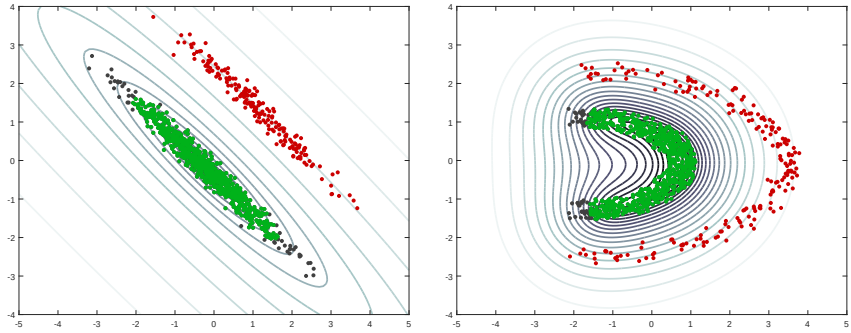


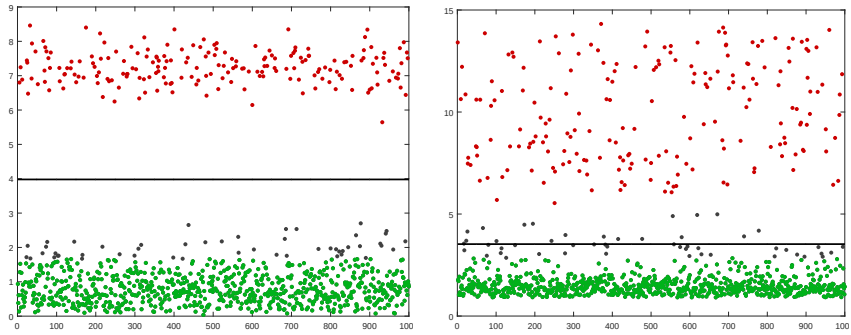
Figure 6.2.2: Results of the non-kernel MCD method on the toy datasets of Figure 6.2.1. The contour lines are level curves of the MCD-based Mahalanobis distance.

are colored red. Since h is lower than the true number of inlying points, some inliers (shown in black) are not included in the h-subset. In the next step, Figure 6.2.3(b) shows the robust Mahalanobis distances, with the horizontal line at the cutoff value given by formula (6.2.15). The final output of KMRCD shown in Figure 6.2.3(c) has the flagged outliers in orange and the points considered inliers in blue. As expected, this result is similar to that of the non-kernel MCD in the left panel of Figure 6.2.2.

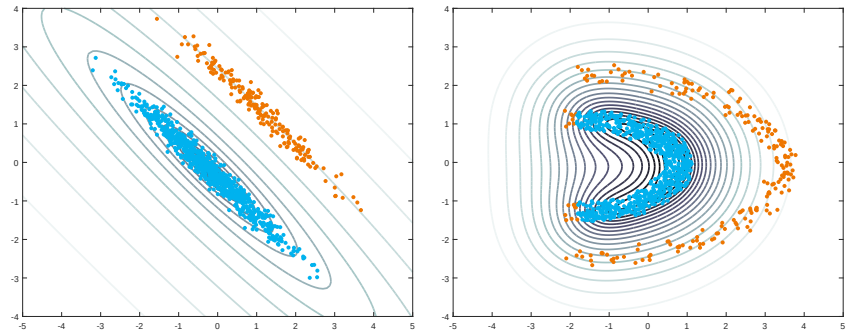
The right hand column of Figure 6.2.3 shows the stages of the KMRCD run on the non-elliptical data. The results for the selected h-subset in Figure 6.2.3(a) look much better than in the right hand panel of Figure 6.2.2, because the level curves of the robust distance now follow the shape of the data. In stage (b) we see that the distances of the inliers and the outliers are fairly well separated by the cutoff (6.2.15), with a few borderline cases, and stage (c) is the final result. This illustrates that using a nonlinear kernel allows us to fit non-elliptical data.



(a) h-subset results after C-step convergence.



(b) Robust Mahalanobis distances with outlier cutoff.



(c) Final results with the flagged outliers shown in orange.

Figure 6.2.3: Kernel MRCD results on the toy datasets of Figure 6.2.1. In the left column the linear kernel was used, and in the right column the RBF kernel. The three stages (a), (b) and (c) are explained in the text.

6.2.5 Simulation study

Simulation study with linear kernel

In this section we compare the KMRCD method proposed in the current section, run with the linear kernel, to the MRCD estimator of Boudt et al. [24]. Recall that using the linear kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}$ means that the feature space can be taken identical to the coordinate space, so using the linear kernel is equivalent to not using a kernel at all. Our purpose is twofold. First, we want to verify whether KMRCD performs well in terms of robustness and accuracy, and that its results are consistent with those of MRCD. And secondly, we wish to measure the computational speedup obtained by KMRCD in high dimensions. In order to obtain a fair comparison we run MRCD with the identity matrix as target, which corresponds to the target of KMRCD. All computations are done in MATLAB on a machine with Intel Core i7-8700K and 16 GB of 3.70GHz RAM.

For the uncontaminated data, that is, for contamination fraction $\varepsilon = 0$, we generate n cases from a d-variate normal distribution with true covariance matrix Σ. Since the methods under consideration are equivariant under translations and rescaling variables, we can assume without loss of generality that the center μ of the distribution is 0 and that the diagonal elements of Σ are all 1. We denote the distribution of the clean data by $\mathcal{N}(0, \Sigma)$. Since the methods are not equivariant to arbitrary nonsingular affine transformations we cannot set Σ equal to the identity matrix. Instead we consider Σ of the ALYZ type, generated as in Section 4 of [8], which yields a different Σ in each replication, but always with condition number 100. The main steps of the construction of Σ in [8] are the generation of a random orthogonal matrix to provide eigenvectors, and the generation of p eigenvalues such that the ratio between the largest and the smallest is 100, followed by iterations to turn the resulting covariance matrix into a correlation matrix while preserving the condition number. (In Appendix G.3.1 also Σ matrices with higher condition numbers were generated, with similar results.)

For a contamination fraction $\varepsilon > 0$ we replace a random subset of $\lfloor \varepsilon n \rfloor$ observations by outliers of different types. *Shift contamination* is generated from $\mathcal{N}(\boldsymbol{\mu}_C, \Sigma)$ where $\boldsymbol{\mu}_C$ lies in the direction where the outliers are hardest to detect, which is that of the last eigenvector $\boldsymbol{v}$ of the true covariance matrix Σ. We rescale $\boldsymbol{v}$ by making $\boldsymbol{v}^T \Sigma^{-1} \boldsymbol{v} = E[Y^2] = d$ where $Y^2 \sim \chi^2_d$. The center is taken as $\boldsymbol{\mu}_C = k\boldsymbol{v}$ where we set $k = 200$. Next, *cluster contamination* stems from $\mathcal{N}(\boldsymbol{\mu}_C, 0.05^2\, \mathbb{I}_d)$ where $\mathbb{I}_d$ is the identity matrix. Finally, *point contamination* places all outliers in the point μ_C so they behave like a tight cluster. These settings make the simulation consistent with those in [24, 112] and [53]. The deviation of an estimated scatter matrix $\hat{\Sigma}$ relative to the

true covariance matrix Σ is measured by the Kullback–Leibler (KL) divergence $\mathrm{KL}(\hat{\Sigma}, \Sigma) = \mathrm{trace}(\hat{\Sigma}\Sigma^{-1}) - \log(\det(\hat{\Sigma}\Sigma^{-1})) - d$. The speedup factor is measured as speedup = time(MRCD)/time(KMRCD). Different combinations of n and d are generated, ranging from $d = n/2$ to $d = 2n$.

Table 6.2.1 presents the Kullback–Leibler deviation results. The top panel is for $\varepsilon = 0$, the middle panel for $\varepsilon = 0.1$ and the bottom panel for $\varepsilon = 0.3$. All table entries are averages over 50 replications. First look at the results without contamination. By comparing the three choices for h, namely $\lfloor 0.5n \rfloor$, $\lfloor 0.75n \rfloor$ and $\lfloor 0.9n \rfloor$, we see that lowering h in this setting leads to increasingly inaccurate estimates $\hat{\Sigma}$. This is the price we pay for being more robust to outliers, since $n - h$ is an upper bound on the number of outliers the methods can handle. When we look at the panels for higher ε we see a similar pattern. When $\varepsilon = 0.1$ the choice $\lfloor 0.9n \rfloor$ is sufficiently robust, and the lower choices of h have higher KL deviation. But when $\varepsilon = 0.3$ only the choice $h = \lfloor 0.5n \rfloor$ can detect the outliers, the other choices cause the estimates to break down. These patterns are confirmed by the averaged $\mathrm{MSE} = \sum_{i=1}^{d} \sum_{j=1}^{d} (\hat{\Sigma} - \Sigma)_{ij}^2 / d^2$ shown in Table G.3.4 in Appendix G.3.3.

From these results we conclude that it is important that h be chosen lower than n minus the number of outliers, but not much lower since that would make the estimates less accurate. A good strategy is to first run with a low h, which reveals the number of outliers, and then to choose a higher h that can still handle the outliers and yields more accurate results as well.

As expected the KMRCD results are similar to those of MRCD, but not identical because there are differences in the selection of initial estimators, also leading to differences in the resulting regularization parameter ρ shown in Table G.3.5 in Appendix G.3.3.

We now turn our attention to the computational speedup factors in Table 6.2.2, that were derived from the same simulation runs as Table 6.2.1. Overall KMRCD ran substantially faster than MRCD, with the factor becoming larger when n decreases and/or the dimension d increases. There are two reasons for the speedup. First of all, the MRCD algorithm computes six initial scatter estimates, of which the last one is the most computationally demanding since it computes a robust bivariate correlation of every pair of variables, requiring $d(d-1)/2$ computations whose total time increases fast with d. Part of the speedup stems from the fact that KMRCD does not use this initial estimator, whereas its own four kernelized initial estimates gave equally robust results. This explains most of the speedup in Table 6.2.2.

For $d > n$ there is a second reason for the speedup, the use of the kernel trick. In particular, each C-step requires the computation of the Mahalanobis distances

Table 6.2.1: Kullback-Leibler deviations of $\hat{\Sigma}$ from Σ.

	Point contamination Value of h/n			Shift contamination Value of h/n			Cluster contamination Value of h/n		
	0.50	0.75	0.90	0.50	0.75	0.90	0.50	0.75	0.90
$\varepsilon = 0$:									
KMRCD									
400×200	126.72	80.03	64.65	127.66	78.99	64.54	129.37	79.42	64.45
300×200	174.37	110.03	88.43	176.46	108.45	87.94	174.52	109.68	87.50
200×200	262.41	171.74	140.18	263.03	172.23	140.66	260.21	169.02	140.45
200×300	492.70	381.13	319.42	491.65	379.07	317.45	491.64	373.38	319.44
200×400	724.41	602.78	535.59	715.59	602.44	532.55	731.76	607.21	537.27
MRCD									
400×200	126.58	80.82	65.65	127.00	79.99	65.63	128.92	80.63	65.35
300×200	175.88	110.57	89.49	176.21	109.22	88.79	174.36	110.28	88.85
200×200	265.04	174.13	141.23	264.57	173.56	141.10	261.93	172.00	141.05
200×300	499.11	384.91	323.48	500.80	383.02	322.14	499.30	378.74	324.27
200×400	734.47	608.04	539.54	729.52	610.79	539.24	738.21	611.83	543.84
$\varepsilon = 0.1$:									
KMRCD									
400×200	128.14	78.48	63.16	127.28	78.28	62.28	128.91	79.75	62.71
300×200	176.76	107.10	86.94	174.32	109.86	87.37	176.13	108.54	87.86
200×200	263.76	172.21	137.31	260.06	171.46	137.72	260.42	171.76	136.27
200×300	493.36	368.80	311.48	488.10	377.46	311.04	491.24	378.81	319.45
200×400	728.07	600.12	558.79	723.40	596.87	535.08	720.44	604.06	534.81
MRCD									
400×200	128.22	79.59	64.19	127.92	79.76	63.48	129.09	81.11	63.91
300×200	174.17	107.67	88.05	172.96	111.15	88.00	173.46	109.50	89.12
200×200	262.71	171.55	137.51	259.52	170.79	138.03	261.48	170.95	136.31
200×300	493.66	368.88	309.88	494.44	382.69	312.85	499.27	382.10	320.48
200×400	723.42	599.88	525.99	736.52	601.15	536.73	733.36	611.43	537.56
$\varepsilon = 0.3$:									
KMRCD									
400×200	127.46	4914.7	2073.9	126.26	1142.7	1613.4	124.73	1124.8	1600.6
300×200	176.51	5104.1	2046.1	176.82	1125.9	1597.6	173.30	1117.2	1555.3
200×200	257.91	5180.6	2038.3	255.55	1168.8	1559.1	257.90	1163.6	1535.9
200×300	485.71	5494.5	2230.0	488.19	1310.0	1626.8	490.05	1311.1	1616.9
200×400	714.57	5779.1	2316.1	721.41	1448.7	1736.7	718.17	1423.4	1721.1
MRCD									
400×200	124.33	6771.6	3082.7	125.15	1395.6	2068.9	124.68	1371.1	2078.5
300×200	164.89	7118.6	3049.4	172.11	1415.2	2076.5	168.51	1393.0	2011.1
200×200	237.08	7519.3	3075.1	241.27	1481.1	2040.1	242.59	1485.8	2014.7
200×300	450.12	8233.1	3413.4	483.07	1653.4	2122.5	483.67	1659.9	2102.4
200×400	663.35	8585.0	3507.7	717.65	1812.8	2212.9	719.74	1790.5	2201.6

Table 6.2.2: Speedup factors of KMRCD relative to MRCD.

	Point contamination Value of h/n			Shift contamination Value of h/n			Cluster contamination Value of h/n		
	0.50	0.75	0.90	0.50	0.75	0.90	0.50	0.75	0.90
$\varepsilon = 0$:									
400×200	97	95	89	99	96	91	100	98	92
300×200	281	242	229	281	242	229	284	242	228
200×200	301	249	227	299	253	226	301	253	228
200×300	661	558	516	623	562	519	657	563	520
200×400	1144	979	897	1157	982	892	1159	978	899
$\varepsilon = 0.1$:									
400×200	98	90	87	96	92	89	96	92	90
300×200	263	227	211	278	240	225	282	239	224
200×200	292	243	216	302	252	225	302	249	228
200×300	631	534	504	652	564	516	664	567	512
200×400	1113	951	870	1157	981	902	1110	980	903
$\varepsilon = 0.3$:									
400×200	77	79	72	100	96	93	99	95	93
300×200	211	193	185	281	251	233	288	251	234
200×200	234	206	202	301	262	238	299	257	238
200×300	543	472	432	653	564	522	654	566	520
200×400	1000	791	749	1161	976	911	1150	977	900

of all cases. MRCD does this by inverting the $d \times d$ covariance matrix $\hat{\Sigma}^H_{\text{reg}}$, whereas KMRCD uses equation (6.2.5) which implies that it suffices to invert the $n \times n$ kernel matrix $\tilde{K}^H_{\text{reg}}$, which takes time complexity $O(n^3)$ instead of $O(d^3)$.

Simulation with nonlinear kernel

In this section we compare the proposed KMRCD estimator to the MRCD estimator of Boudt et al. [24] on two types of non-elliptical datasets. The first type is generated by a copula. We start by considering the t copula [167] with Pearson correlation 0.1 and $\nu = 1$ degrees of freedom. The black points in the left panel of Figure 6.2.4 were generated from this copula. We then added contamination in the form of uniformly distributed random noise on the unit square, where points lying close to the regular distribution were removed. The red points in the left panel of Figure 6.2.4 are the outliers. Apart from the t copula we also consider the Frank, Clayton, and Gumbel copulas with Kendall

rank correlation $\tau = 0.6$. They are visualized in Figure 9 in the Supplementary Material.

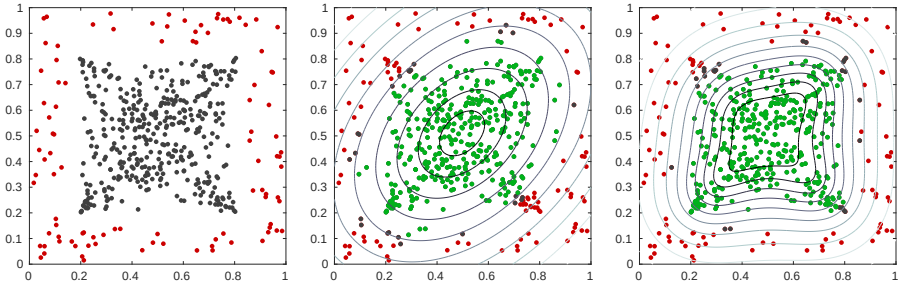


Figure 6.2.4: Illustration of the non-elliptical simulation setting with data generated from the t copula, plus 20% of outlying observations. In the left panel, the regular observations are shown in black and the outliers in red. The results of the MRCD estimator are in the middle panel, and those of the KMRCD estimator in the rightmost panel, each for $h = 0.75n$. In those panels the points in the h-subset are shown in green, and the other points with the $n(1-\varepsilon)$ lowest (kernel) Mahalanobis distance are depicted in grey. The remaining points are shown in red. The curves are contours of the robust (kernel) Mahalanobis distance.

The proposed estimator is also benchmarked in a second type of setting where the regular observations are uniformly distributed on the unit circle, and inside the circle are outliers generated from the Gaussian distribution with center 0 and covariance matrix equal to 0.04 times the identity matrix. This setting is illustrated in Figure 6.2.5. This is a simple example where the clean data lie near a manifold.

In the simulation we generated 100 datasets of each type, with $n = 500$ and the outlier fraction ε equal to 0.1 or 0.2, so the number of regular observations is $n(1-\varepsilon)$. With all four copulas the KMRCD estimator used the radial basis function with bandwidth (6.2.16). For the circle-based data the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + 1)^2$ of degree 2 was used.

We measure the performance by counting the number of outliers in the h-subset, and among the $n(1-\varepsilon)$ points with the lowest (kernel) Mahalanobis distance. The averaged counts over the 100 replications are shown in Table 6.2.3. By comparing the rows of KMRCD and MRCD with the same ε, we see that MRCD has more true outliers in its h-subset and its $n(1-\varepsilon)$ set. In the table, KMRCD outperforms MRCD for both choices of ε and for all three choices of h. The good performance of KMRCD is also seen in the right panel of Figure 6.2.4, where the

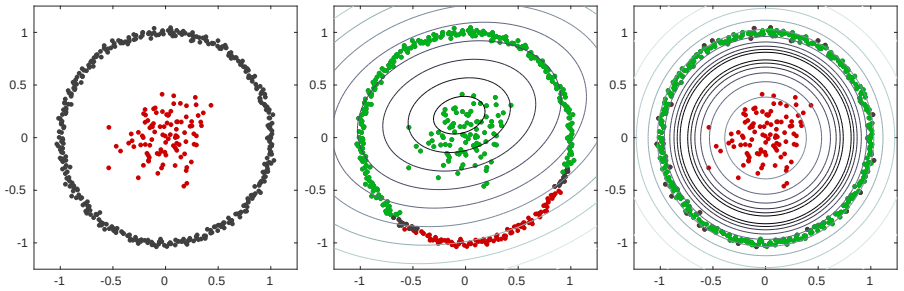


Figure 6.2.5: Illustration of the non-elliptical simulation setting with data generated from the circle manifold, plus 20% of outlying observations. The remainder of the description is as in Figure 6.2.4.

Table 6.2.3: Average number of outliers in the h-subset H, and among the $n(1-\varepsilon)$ points with lowest (kernel) Mahalanobis distance.

	t copula			Frank copula			Clayton copula			Gumbel copula			Circle		
	0.75	0.8	0.9	0.75	0.8	0.9	0.75	0.8	0.9	0.75	0.8	0.9	0.75	0.8	0.9
$\varepsilon = 0.1$:															
KMRCD															
H	0	0	0.5	0	0	2.9	0	0	2.6	0	0	2.8	0	0	0
$n(1-\varepsilon)$	10	8.2	0.5	3.9	4.2	2.9	4.3	4.4	2.6	4.4	4.7	2.8	0	0	0
MRCD															
H	1.2	2.9	13.3	1.3	2.1	11.2	2.4	3.7	11.6	2.0	2.9	11.2	50	50	50
$n(1-\varepsilon)$	22.5	20.3	13.3	12.9	12.2	11.2	14.4	14.0	11.6	13.7	13.3	11.2	50	50	50
$\varepsilon = 0.2$:															
KMRCD															
H	0.1	2.1	/	0	3.5	/	0	3.3	/	0	3.8	/	0	0	/
$n(1-\varepsilon)$	9.0	2.1	/	4.8	3.5	/	4.4	3.3	/	5.2	3.8	/	0	0	/
MRCD															
H	12.9	21.7	/	8.4	16.3	/	10.2	18.5	/	10	18.7	/	100	100	/
$n(1-\varepsilon)$	27.5	21.7	/	18.9	16.3	/	19.7	18.5	/	20.1	18.7	/	100	100	/

contours of the kernel Mahalanobis distance nicely follow the distribution. The difference between MRCD and KMRCD is most apparent on the circle-based data: in Figure 6.2.5 the KMRCD fits the regular data on the circle, whereas the original MRCD method, by its nature, considers the outliers in the center as regular data.

We conclude that in this nonlinear setting, KMRCD has successfully extended the MRCD to non-elliptical distributions. We want to add two remarks about this. First, as in all kernel-based methods the choice of the kernel is important, and choosing a different kernel can lead to worse results. And second, just as in

the linear setting h should be lower than n minus the number of outliers, so in practice it is recommended to first run with a low h, look at the results in order to find out how many outliers there are, and possibly run again with a higher h.

Appendix G.3.2 contains additional simulation results about the computation time of the four initial estimators in KMRCD and their subsequent C-steps, in different settings with linear and nonlinear kernels.

6.2.6 Experiments

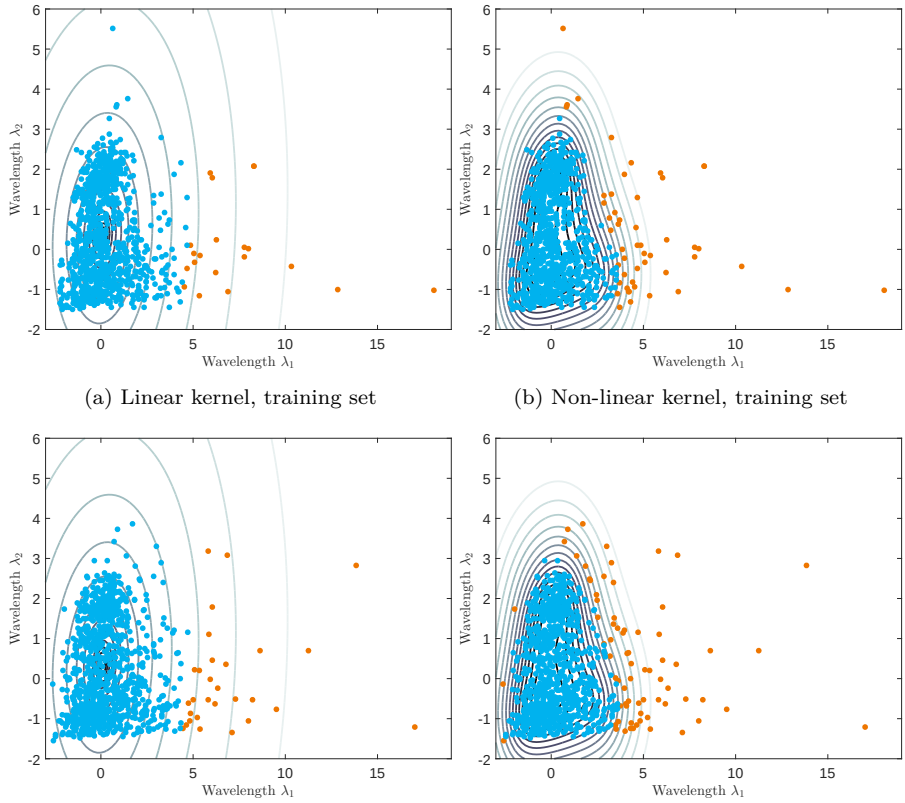
Food industry example

We now turn our attention to a real dataset from the food industry. In that setting datasets frequently contain outliers, because samples originate from natural products which are often contaminated by insect damage, local discolorations and foreign material. It also happens that the image acquisition signals yield non-elliptical data, and in that case a kernel transform can help.

The dataset is bivariate and contains two color signals measured on organic sultana raisin samples. The goal is to classify these into inliers and outliers, so that during production outliers can be physically removed from the product in real time. There are training data and test data, but the class label 'outlier' is not known beforehand. The scatter plot of the training data in Figure 6.2.6 (a) reveals the non-elliptical (and to some extent triangular) structure of the inliers. Three types of outliers are visible. Those with high values of λ_1 and low λ_2 at the bottom right correspond to foreign, cap-stem related material like wood, whereas points with high values of λ_2 represent discolorations. There are also a few points with high values of both λ_1 and λ_2 which correspond to either discolored raisins or objects with clear attachment points of cap-stems. Outliers of any of these three types need to be flagged and removed from the product. From manually analyzing data of this product it is known beforehand that the fraction of outliers is rather low, at most around 2%.

We first run KMRCD on the training data. In its preprocessing step it standardizes both variables. For comparison purposes we use two kernels. In the left hand column of Figure 6.2.6 we apply the linear kernel, and in the right hand column we use the RBF kernel with tuning constant σ given by (6.2.16). Since we know the fraction of outliers is low we can put $h = \lfloor 0.95n \rfloor$. Each figure shows the flagged points in orange and the remaining points in blue, and the contour lines are level curves of the robust distance.

The fit with linear kernel in Figure 6.2.6 (a) has contour lines that do not follow the shape of the data very well, and as a consequence it fails to flag some of



(a) Linear kernel, training set

(b) Non-linear kernel, training set

(c) Linear kernel, test set

(d) Non-linear kernel, test set

Figure 6.2.6: Food industry example: KMRCD results with the linear kernel (left column) and the RBF kernel (right column). The top row contains the training data, and the resulting fits were applied to the test data in the bottom row.

the outliers, such as those with high λ_2 and some with relatively high values of both λ_1 and λ_2. The KMRCD fit with nonlinear kernel in Figure 6.2.6 (b) has contour lines that model the data more realistically. This fit does flag all three types of outliers correctly. Both trained models were then used to classify the previously unseen test set. The results are similar to those on the training data. The anomaly detection with linear kernel in Figure 6.2.6 (c) again misses the raisin discolorations, which would keep these impurities in the final consumer product. Fortunately, the method with the nonlinear kernel in panel (d) does flag them.

MNIST digits data

Our last example is high dimensional. The MNIST dataset contains images of handwritten digits from 0 to 9, at the resolution of 28×28 grayscale pixels (so there are 784 dimensions), and was downloaded from `http://yann.lecun.com/exdb/mnist`. There is a training set and a test set. Both were subsampled to 1000 images. To the training data we added noise distributed as $\mathcal{N}(0, (0.5)^2)$ to 20% of the images, and in the test set we added noise with the same distribution to all images. We then applied KMRCD with RBF kernel with tuning constant σ given by (6.2.16) and $h = \lfloor 0.75n \rfloor$ to the 1000 training images. Next, we computed the eigenvectors of the robustly estimated covariance matrix.

Our goal is to denoise the images in the test set by projecting them onto the main eigenvectors found in the training data. As we are interested in a reconstruction of the data in the original space rather than in the feature space, we transform the scores back to the original input space by the iterative optimization method of [160].

The top panel of Figure 6.2.7 illustrates what happens when applying this computation to the classical covariance matrix in feature space, which corresponds to classical kernel PCA [208]. The bottom panel is based on KMRCD. The first row of each panel displays original test set images, and the second row shows the test images after the noise was added. The first and second rows do not depend on the estimation method, but the remaining rows do. There we see the results of projecting on the first 5, 15, and 30 eigenvectors of each method. In the top panel those images are rather diffuse, which indicates that the classical approach was affected by the training images with added noise and considers the added noise as part of its model. This implies that increasing the number of eigenvectors used will not improve the overall image quality much. The lower panel contains sharper images, because the robust fit of the training data was less affected by the images with added noise that acted as outliers.

We can also compute the mean absolute error $\sum_{i=1}^{n} \sum_{j=1}^{d} |\boldsymbol{x}_{i,d} - \hat{\boldsymbol{x}}_{i,d}|/(nd)$ between the original test images (with $d = 784$ dimensions) and the projected versions of the test images with added noise. Figure 6.2.8 shows this deviation as a function of the number of eigenvectors used in the projection. The deviations of the robust method are systematically lower than those of the classical method, confirming the visual impression from Figure 6.2.7.

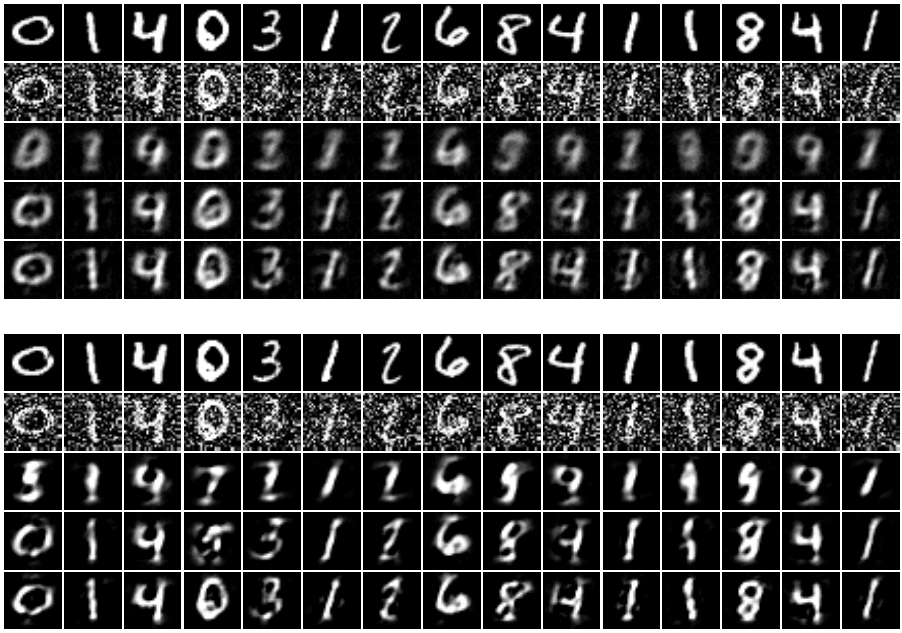


Figure 6.2.7: MNIST denoising results based on the classical covariance matrix (top panel) and on the KMRCD estimate (bottom panel). The first and second rows of each panel show the same original and noise-added test set images. The remaining rows contain the results of projecting on the first 5, 15, and 30 eigenvectors.

6.3 Conclusion

In Section 6.1, we robustified the Gen-RKM method. Using a weighted conjugate feature duality, an RKM formulation for weighted kernel PCA is proposed. This formulation is used within the Gen-RKM training procedure to fine-tune the latent space using a weighting function based on the MCD. Experiments show that the weighted RKM is capable of generating denoised images in spite of contamination in the training data. Furthermore, being a latent variable model, robust Gen-RKM preserves the disentangled representation. In Section 6.2, the minimum regularized covariance determinant method (MRCD), which is a robust estimator for multivariate location and scatter, is generalized to a kernel-induced feature space. The kernel MRCD method introduced in this chapter is a robust method that allows to analyze non-elliptical data when used with a nonlinear kernel. Another advantage is that even when using the linear kernel the computation becomes much faster when there are more dimensions

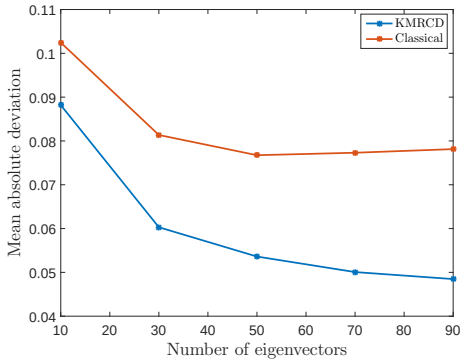


Figure 6.2.8: The mean absolute error of the denoised images to the original test images in function of the number of eigenvectors used in the projection. The top curve is for the classical covariance matrix in feature space, the lower curve for KMRCD.

than cases, a situation that is quite common nowadays. Due to the built-in regularization the result is always well-conditioned. The algorithm starts from four kernelized initial estimators, and to each it applies a new kernelized refinement step. The remainder of the algorithm is based on a theorem showing that C-steps in feature space are equivalent to a new type of C-steps on centered kernel matrices, so the latter reduce the objective function. The performance of KMRCD in terms of robustness, accuracy and speed is studied in a simulation, and the method is applied to several examples.

Chapter 7

Conclusion

Contributions

In this thesis, we have developed and analyzed multiple methods to improve trustworthiness in kernel methods. The algorithms are introduced within three major themes: diversity (Chapters 2-4), disentanglement (Chapter 5), and robustness (Chapter 6). In the following, we give a chapter-by-chapter overview of our contributions.

In Chapter 2, the interplay between diversity and regularization is analyzed. By the use of DPPs for sampling, multiple theoretical results are obtained. Experiments show that diverse sampling methods improve performance in several kernel methods compared to uniform sampling. This is demonstrated in the Nyström approximation, KPCA, and kernel ridge regression. In the context of KRR, we proposed the use of the RLS distribution to assess the uniformity of the performance of the regressor. Methods trained with diversity sampling are shown to have a more uniform performance over the data, hence reducing bias.

In Chapter 3, three new methods that promote diversity sampling are introduced. The methods DAS and RAS are motivated by the connection between leverage scores, DPPs, and kernelized Christoffel functions. An additional approximation is proposed so that RAS can be applied to large data sets in the case of a Gaussian kernel. A theoretical analysis is provided along with an experimental section that analyzes the accuracy of the method in the Nyström approximation. Experiments for kernel ridge regression show that the proposed method performs much better in the tail of the data while maintaining comparable performance for the bulk data, thus reducing bias in the model. The third method is a greedy

deterministic adaptation of k-DPP. The proposed method is evaluated by the accuracy of the Nyström approximation on multiple datasets. Experiments show that the proposed method can increase diversity, and improve performance. The improvement is the most notable when there is a fast decay of the eigenvalues of the kernel matrix.

In Chapter 4, BuresGAN is introduced. An additional term based on the Bures distance which promotes matching of the distribution of the generated and real data is added to the objective. Experiments show that the proposed method reduces mode collapse and improves sample quality. When there are minority modes in the data, RLS sampling for training GANs is introduced. RLS sampling is easy to integrate into any GAN model with only minimal changes to the training procedure. Three feature maps are proposed: an implicit feature map, a fixed explicit feature map, and the discriminator network as a feature extractor. Two approximation methods for the explicit feature maps are also discussed. Experiments show that RLS sampling in GANs results in a more uniform sampling over all the modes, regardless of the original weight of the mode in the data distribution.

In Chapter 5, generative restricted kernel machines are proposed. When using kernel functions, the training consists of only solving an eigenvalue problem. In the case of (convolutional) neural network-based explicit feature maps, a training procedure that involves joint feature selection and subspace learning is proposed. Moreover, the RKM framework is extended to multi-view generation. The training is made scalable to large datasets by restricting the interconnection matrix U to be valued on a Stiefel manifold and stochastic training using the Cayley Adam algorithm of [138]. Experiments on benchmark datasets illustrate the merit of the proposed framework for generation quality as well as disentanglement. Additionally, we establish connections with probabilistic models, formulate an Evidence Lower Bound (ELBO), and discuss the independence of latent factors.

In Chapter 6, two robust methods are proposed. A robust Gen-RKM method is formulated where the latent space is weighted based on the minimum covariance determinant method. Potential outliers are down-weighted, and therefore not encoded in the representation of the model. Experiments show that the weighted RKM is capable of generating denoised images despite contamination in the training data. Furthermore, being a latent variable model, robust Gen-RKM preserves the disentangled representation. Second, the MRCD estimator is generalized to the kernel domain. The proposed method allows the analysis of non-elliptical data when used with a nonlinear kernel. Furthermore, when using the linear kernel, the computation becomes much faster when there are more dimensions than cases, a situation that is quite common nowadays. The performance of KMRCD in terms of robustness, accuracy, and speed is studied

in a simulation study, and the method is applied to several examples.

Future work

The remaining part of this thesis provides some suggestions regarding prospective work.

- *Diversity sampling in imbalanced classification problems.* Frequently data consists of a majority and a minority class which makes up a very small percentage of the data. For example in fraud data, fraud occurs typically in less than 0.5% of the cases. Classifying correctly in such a highly imbalanced data set typically leads to predictions that favor the majority group, causing fraud to remain undetected. A common approach to deal with imbalanced data is random undersampling of the majority class. However, uniform sampling can lead to a lack of diversity, or the full space of the data not being fully represented. A potential research direction is the effect of different diverse sampling methods for undersampling imbalanced classification problems.

- *Diversity sampling for training neural networks.* The effect of the batch size and quality of the batch on the training of deep or wide neural networks is currently actively studied [39, 257]. A potential research direction is the effect of different sampling methods for the mini-batches used in stochastic optimization. In particular, the role of the diversity of the minibatch could be studied in connection with the role of implicit regularisation.

- *Deep level-wise Gen-RKM.* The proposed Gen-RKM model makes use of a deep neural network-based feature map. This allows the modeling of complex data such as images. In [224], deep RKMs are proposed in the level sense by coupling multiple feature extractors in a cascade. Future work would consist of combining and cascading multiple feature extractors into one model.

- *Gen-RKM and learning with few labels.* The Gen-RKM model is currently only used for unsupervised learning. In several applications, it is not uncommon that the label of a few data points is known. In the context of disentanglement, this label information corresponds to underlying generating factors of the representation you want to learn. The inclusion of this information could improve the quality of the disentanglement representation. A similar idea was already proposed in the context of VAEs [147].

- *Gen-RKM and adversarial robustness.* A recent publication [34] has formalized what it means for a VAE to be adversarially robust. A novel criterion for robustness in probabilistic models called r-robustness is proposed. This is used to construct the first theoretical results for the robustness of VAEs. The authors derive margins in the input space for which they can provide guarantees about the resulting reconstruction. Informally, a region is defined within which any perturbation will produce a reconstruction that is similar to the original reconstruction. In section 5.3, a stochastic autoencoder loss is proposed to the objective function which adds a small noise term to the latent space. The intuition is that this noise term gives a smoother decoder network, hence a more robust latent space against adversarial attacks. The interplay between this loss and the aforementioned robustness criterion would be interesting to explore.

- *Robust kernel methods by kernel MRCD.* Many linear methods were successfully robustified using the MCD estimator, such as linear regression [198], PCA [111], discriminant analysis [113],... Through the extension of the linear MRCD to the kernel setting in [211], a similar robustification can be achieved for the kernel variants of the above methods.

Appendix A

Diversity sampling is an implicit regularization for kernel methods

This appendix contains the supplementary material of the work presented in Section 2.1. In Section A.1, the details about our theoretical results are given. Next, information on the parameters and used datasets is given in Section A.2. Additional experiments are described in Section A.3.

A.1 Proofs

Proof of Theorem 2.1.1. Without loss of generality, we put $\alpha = 1$.

(i) Then, we first prove $\mathbb{E}_{\mathcal{C}}[CK_{\mathcal{C}\mathcal{C}}^{-1}C^{\top}] = (K+\mathbb{I})^{-1}$. The matrix inversion lemma yields

$$\begin{aligned}\det\begin{pmatrix} K_{\mathcal{C}\mathcal{C}} & C^{\top}\boldsymbol{w} \\ \boldsymbol{u}^{\top}C & 1 \end{pmatrix} &= \det(K_{\mathcal{C}\mathcal{C}})\left[1 - \boldsymbol{u}^{\top}CK_{\mathcal{C}\mathcal{C}}^{-1}C^{\top}\boldsymbol{w}\right] \\ &= \det\left(K_{\mathcal{C}\mathcal{C}} - C^{\top}\boldsymbol{w}\boldsymbol{u}^{\top}C\right).\end{aligned}$$

This simplifies to

$$\boldsymbol{u}_{\mathcal{C}}^{\top}K_{\mathcal{C}\mathcal{C}}^{-1}\boldsymbol{w}_{\mathcal{C}} = 1 - \frac{\det(K_{\mathcal{C}\mathcal{C}} - C^{\top}\boldsymbol{w}\boldsymbol{u}^{\top}C)}{\det(K_{\mathcal{C}\mathcal{C}})}.$$

By taking the expectation on both sides, we find

$$\mathbb{E}_{\mathcal{C}}\left[\boldsymbol{u}_{\mathcal{C}}^{\top} K_{\mathcal{C}\mathcal{C}}^{-1} \boldsymbol{w}_{\mathcal{C}}\right] = 1 - \frac{\det(\mathbb{I} + K - \boldsymbol{w}\boldsymbol{u}^{\top})}{\det(\mathbb{I} + K)} = \boldsymbol{u}^{\top}(\mathbb{I} + K)^{-1}\boldsymbol{w},$$

where we used $\sum_{\mathcal{C}\subseteq[n]} A_{\mathcal{C}\mathcal{C}} = \mathbb{I} + A$ with A a square matrix.

(ii) Secondly, we prove that $\mathbb{E}_{\mathcal{C}}[\boldsymbol{u}^{\top} CC^{\top} KCC^{\top}\boldsymbol{v}] = P_{(2)} \circ K$. We first calculate by using the marginal kernel

$$\mathbb{E}_{\mathcal{C}}[1_{i\in\mathcal{C}} 1_{j\in\mathcal{C}}] = 1_{i=j} \times P_{ii} + 1_{i\neq j} \times \det P_{\{i,j\}\{i,j\}} = 1_{i=j} \times P_{ii} + (P_{ii}P_{jj} - P_{ij}^2).$$

The result follows from the following formula

$$\mathbb{E}_{\mathcal{C}}[\boldsymbol{u}^{\top} CC^{\top} KCC^{\top}\boldsymbol{v}] = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}_{\mathcal{C}}[u_i 1_{i\in\mathcal{C}} K_{ij} 1_{j\in\mathcal{C}} v_j],$$

and by using the linearity of the expectation. □

Proof of Corollary 2.1.1. We follow the proof strategy of [136]. Let the following $f(\mathcal{C}) = \boldsymbol{w}_{\mathcal{C}}^{\top} K_{\mathcal{C}\mathcal{C}}^{-1} \boldsymbol{w}_{\mathcal{C}}$ a function viewed as $f : \{0,1\}^n \to \mathbb{R}$. Firstly, we use a concentration inequality, then we upper bound the Lipschitz constant of f.
Concentration inequality. To start with, let us denote by N the number of points sampled by a DPP. Then, we quote a concentration result from [18].

Theorem A.1.1 (Thm 3.5 in [178])**.** *Let a determinantal point process with $\mathbb{E}N = \mu < \infty$. Let f be a 1-Lipschitz function on finite counting measures. Then ,*

$$\Pr(|f - \mathbb{E}f| > a) \leq 5\exp\left(-\frac{a^2}{16(a+2\mu)}\right).$$

Let $\delta > 0$ be an upper bound on the failure probability, such that $5\exp\left(-\frac{a^2}{16(a+2\mu)}\right) \geq \delta$ for some $a > 0$. This yields to the inequality

$$a^2 - 16a\log\left(\frac{5}{\delta}\right) - 32\mu\log\left(\frac{5}{\delta}\right) \geq 0.$$

Therefore, we find the lower bound

$$a \geq 8\left(\log\left(\frac{5}{\delta}\right) + \sqrt{\left(\log\left(\frac{5}{\delta}\right)\right)^2 + \frac{\mu}{2}\log\left(\frac{5}{\delta}\right)}\right). \tag{A.1.1}$$

Then, notice that we can use the above Theorem for ℓ-Lipshitz function by simply replacing f by f/ℓ. Hence, with a probability of at least $1-\delta$, we have

$$\begin{aligned}|f - \mathbb{E}f| \leq & \ell 8 \left(\log\left(\frac{5}{\delta}\right) + \sqrt{\left(\log\left(\frac{5}{\delta}\right)\right)^2 + \frac{\mu}{2}\log\left(\frac{5}{\delta}\right)} \right) \\ & \leq 8\ell \left(2\log\left(\frac{5}{\delta}\right) + \sqrt{\frac{\mu}{2}\log\left(\frac{5}{\delta}\right)} \right) \\ & \leq 8\ell \left(2 + \sqrt{\frac{\mu}{2}} \right) \log\left(\frac{5}{\delta}\right),\end{aligned}$$

where we used $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for positive a and b, and that $\log\left(\frac{5}{\delta}\right) \geq 1$.

Bounding the Lipschitz constant. We now calculate an upper bound on the Lipschitz constant of f with respect to the Hamming distance. To do so, it is sufficient to consider two binary vectors C and $C' \in \{0,1\}^n$ differing of one digit. Say that $f(C) \geq f(C') \geq 0$. Then, we have

$$|f(C) - f(C')| \leq f(C) - f(C') \leq \boldsymbol{w}_{\mathcal{C}}^\top K_{\mathcal{C}\mathcal{C}}^{-1} \boldsymbol{w}_{\mathcal{C}} \leq \frac{1}{\lambda_{\min}(K)} = \frac{d_H(C,C')}{\lambda_{\min}(K)},$$

since $\|\boldsymbol{w}\|_2 = 1$ and where $d_H(C,C')$ is the Hamming distance between the binary vectors C and C'. Notice that the inequality above used the following fact

$$K_{\mathcal{C}\mathcal{C}}^{-1} \preceq \frac{1}{\lambda_{\min}(K_{\mathcal{C}\mathcal{C}})} \mathbb{I}_{\mathcal{C}\mathcal{C}},$$

and $\lambda_{\min}(K) \leq \lambda_{\min}(K_{\mathcal{C}\mathcal{C}})$ (see [115]), as well as $CC^\top \preceq \mathbb{I}$. The final result follows by taking $\ell = 1/\lambda_{\min}(K)$. □

Proof of Theorem 2.1.2. Since X is a orthogonal projector and $\hat{K} \preceq K$ with Nyström approximation $\hat{K} = \hat{K}(K, \mathcal{C})$, we have $\operatorname{Tr}(X\hat{K}X) \leq \operatorname{Tr}(XKX)$, which yields

$$\operatorname{Tr}(K - XKX) \leq \operatorname{Tr}(\hat{K} - X\hat{K}X) + \operatorname{Tr}(K - \hat{K}) = \operatorname{Tr}(K) - \operatorname{Tr}(X\hat{K}X).$$

By taking the expectation over $\mathcal{C} \sim DPP_L(K/\alpha)$ on both sides of the above inequality, and by using $\mathbb{E}_{\mathcal{C}}(\hat{K}) = K - \alpha P$, we obtain

$$\operatorname{Tr}(K - XKX) \leq \mathbb{E}_{\mathcal{C}}[\operatorname{Tr}(\hat{K} - X\hat{K}X)] + \alpha d_{\text{eff}}(K/\alpha) = \operatorname{Tr}(K - XKX) + \alpha \operatorname{Tr}(XPX).$$

Finally, by using that X is a projector of rank k, it holds that $\operatorname{Tr}(XPX) \leq \operatorname{Tr}(X) = k$ since $P \preceq \mathbb{I}$. Also, we have $\operatorname{Tr}(XPX) \leq \operatorname{Tr}(P) = d_{\text{eff}}(K/\alpha)$, so that the final bound is obtained. □

Proof of Theorem 2.1.3. The risk $\mathcal{R}(\hat{z}) = \frac{1}{n}\mathbb{E}_\epsilon\|\hat{z} - z\|_2^2$ is decomposed in terms of bias and variance $\mathcal{R}(\hat{z}) = \text{bias}^2(K) + \text{var}(K)$, where the bias reads

$$\text{bias}(K) = \sqrt{n\gamma^2 z^\top (K + n\gamma\mathbb{I})^{-2} z}$$

and the variance is

$$\text{var}(K) = \frac{\sigma^2}{n}\,\text{Tr}[K^2(K + n\gamma\mathbb{I})^{-2}].$$

Firstly, since $\hat{K} \preceq K$, it holds that $\lambda_\ell(\hat{K}) \leq \lambda_\ell(K)$ for all ℓ and $\text{var}(\hat{K}) \preceq \text{var}(K)$. Then, we can give an upper bound for $\text{bias}(\hat{K})$. The 2-norm submultiplicativity gives

$$\begin{aligned}\|(\hat{K} + n\gamma\mathbb{I})^{-1}z - (K + n\gamma\mathbb{I})^{-1}z\|_2 &\leq \|(\hat{K} + n\gamma\mathbb{I})^{-1}(K - \hat{K})\|_2 \|(K + n\gamma\mathbb{I})^{-1}z\|_2 \\ &\leq \frac{\text{Tr}(K - \hat{K})}{n\gamma}\|(K + n\gamma\mathbb{I})^{-1}z\|_2.\end{aligned}$$

Hence, by using the triangle inequality and the bound hereabove, it holds that

$$\begin{aligned}\sqrt{z^\top(\hat{K} + n\gamma\mathbb{I})^{-2}z} &\leq \|(K + n\gamma\mathbb{I})^{-1}z\|_2 + \|(\hat{K} + n\gamma\mathbb{I})^{-1}z - (K + n\gamma\mathbb{I})^{-1}z\|_2 \\ &\leq (1 + \frac{\text{Tr}(K - \hat{K})}{n\gamma})\sqrt{z^\top(K + n\gamma\mathbb{I})^{-2}z}.\end{aligned}$$

The result follows by taking the expectation over $\mathcal{C} \sim DPP_L(K/\alpha)$. □

Proof of Lemma 2.1.1. This result is a direct consequence of the two following identities: $\mathbb{E}_\mathcal{C}[1_{i\in\mathcal{C}}] = P_{ii}$, and

$$\mathbb{E}_\mathcal{C}[1_{i\in\mathcal{C}}1_{j\in\mathcal{C}}] = 1_{i=j} \times P_{ii} + (P_{ii}P_{jj} - P_{ij}^2).$$

□

A.2 Parameters and dataset descriptions

The parameters and datasets used in the simulations can be found in Table A.2.1. When a subset is sampled from k-DPP, the number of landmarks is fixed to $k = d_{\text{eff}}(n\lambda)$, where the effective dimension corresponds to the expected subset size for $DPP_L(K/(n\lambda))$.

Table A.2.1: Datasets and parameters used for the experiments on the Nyström approximation.

Dataset	Task	n	d	σ	λ	$d_{\text{eff}}(n\lambda)$	n_{RRLS}
Housing	Kernel approx.	506	13	5	10^{-6}	186	/
MiniBooNE	Kernel approx.	130065	50	8	10^{-6}	462	8000
codRNA	Kernel approx.	331152	8	8	10^{-6}	1204	8000
B. Cancer	KPCA	569	30	10	10^{-6}	158	/
A. Credit	KPCA	690	14	5	10^{-6}	371	/
Adult	KPCA	48842	110	8	10^{-6}	1202	8000
Covertype	KPCA	581012	54	20	10^{-6}	3665	10000
Abalone	KRR	4177	8	1	10^{-4}	294	/
Wine Quality	KRR	6497	11	2	10^{-4}	555	/
Bike S.	KRR	17389	16	1	10^{-4}	2732	8000
Year Pred.	KRR	515345	90	10	10^{-4}	4217	10000
Parkinson	Prec.	5875	20	5	10^{-6}	738	/
Pumadyn8FM	Prec.	8192	25	5	10^{-6}	296	/

A.3 Additional experiments

Setting. In the performance plots displayed in the sequel, i.e., Figures A.3.2, A.3.3, A.3.4, A.3.5, A.3.6, A.3.7 and A.3.8, the results are plotted on a logarithmic scale, averaged over 10 trials, and the error bars show the 0.05 and 0.95 quantile. Recall that the larger the $\log(\det(K_{\mathcal{CC}}))$, the more diverse the subset. The computer used for the small-scale simulations has 8 cores 3.40GHz and 15.5 GB of RAM. Large scale experiments on Covertype and Year PredictionMSD were done on the Vlaams Super Computer (VSC). The implementation of the algorithms is done with matlabR2018b.

Additional case study: Clustering. The performance of diverse kernel approximation methods is evaluated for a clustering task using kernel k-means with Nyström approximation [240]. Samples with different diversities are sampled, afterwards *Algorithm 2* of Wang et al. [240] is used to cluster the dataset, where the target dimension $s = k$ is always equal to the number of desired clusters. The clustering performance is evaluated by the normalized mutual information (NMI) [221], the NMI gives a value between 0 and 1, where 1 represents a perfect correlation between the ground truth and the clustering outcome. We first illustrate the effect of using a diverse sampling on a toy example. In Figure A.3.1, we show a very imbalanced dataset consisting of

Table A.3.1: Datasets and parameters used in the clustering experiments.

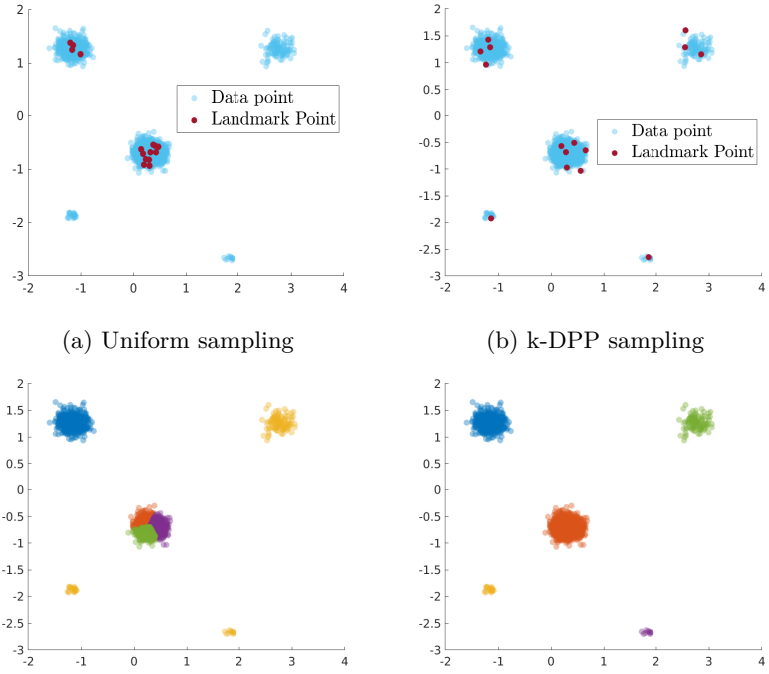
Dataset	n	d	σ	λ	$d_{\text{eff}}(n\lambda)$	# Clusters
`Glass`	214	9	9	10^{-4}	24	7
`Cancer`	569	30	3	10^{-4}	363	2
`Credit`	690	14	2	10^{-4}	456	2

5 Gaussian bumps with a different number of points and different variances. Uniform sampling often only selects landmarks from 3 out of the 5 clusters, whereas a k-DPP samples landmarks out of every cluster. Consequently, the kernel k-means algorithm with Nyström approximation can be improved by using diverse sampling. It is important to note that the superior performance is due to the histogram of ridge-leverage scores with $\gamma = 10^{-4}$ (a measure of outlyingness) having a long tail together with the different clusters being heavily imbalanced.

Next, we demonstrate the effect of diverse sampling on the `Glass`, `Breast Cancer` and `Australian Credit` datasets[1] of Table A.3.1. The condition number, smallest/largest eigenvalues of $K_{\mathcal{CC}}$ and NMI is plotted as a function of the determinant. The averaged results are visualized in Figure A.3.2. Information on the datasets and hyperparameters used for the experiments is given in Table A.3.1. Sampling a diverse subset gives a more accurate clustering. Similar as to the kernel approximation experiments, in the presence of outliers, taking samples with an extremely large $\det(K_{\mathcal{CC}})$ thanks to the Greedy Swapping Algorithm might decrease the accuracy as for the `Glass` dataset.

Supplementary numerical experiments. Several additional illustrations, obtained with the main methodology as in the manuscript, on the datasets given in Table A.3.2 are given in the sequel. Nyström approximation error is given in Figures A.3.3 and A.3.8, Kernel PCA in Figure A.3.5, and Kernel Ridge Regression in Figures A.3.6 and A.3.7. In most of the tasks illustrated in those figures, a larger diversity yields an improved performance. Let us discuss some particular cases.

Kernel approximation. As it was mentioned already hereabove, in the presence of outliers, a very diverse subsample can produce a poor kernel approximation. This can be viewed in Figure A.3.3i, where the greedy algorithm is able to select subsets with a very large diversity. The randomized sampling methods that we studied empirically here did not suffer from this issue.



(a) Uniform sampling

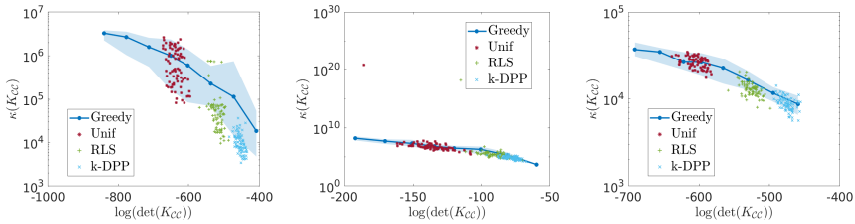
(b) k-DPP sampling

(c) Clustering: Uniform

(d) Clustering: k-DPP

Figure A.3.1: Illustration of sampling methods on an artificial clustering problem. Uniform sampling oversamples dense parts and does not select landmark points in every cluster. k-DPP sampling overcomes this limitation, resulting in better clustering performance.

KRR. By using the same methodology as in the manuscript, the Mean Absolute Percentage Error is calculated both in the bulk and in the tail of the leverage score distribution of the test set. Notice that uniform sampling can often reduce the MAPE in the bulk of the data, while diverse sampling yields a larger improvement in the tail of the distribution (cfr. Figure A.3.6e). Note that the difference between Uniform sampling and RLS sampling is less apparent, this may be caused by the shape of the RLS distribution (see Figure A.3.7), which does not have a long tail.



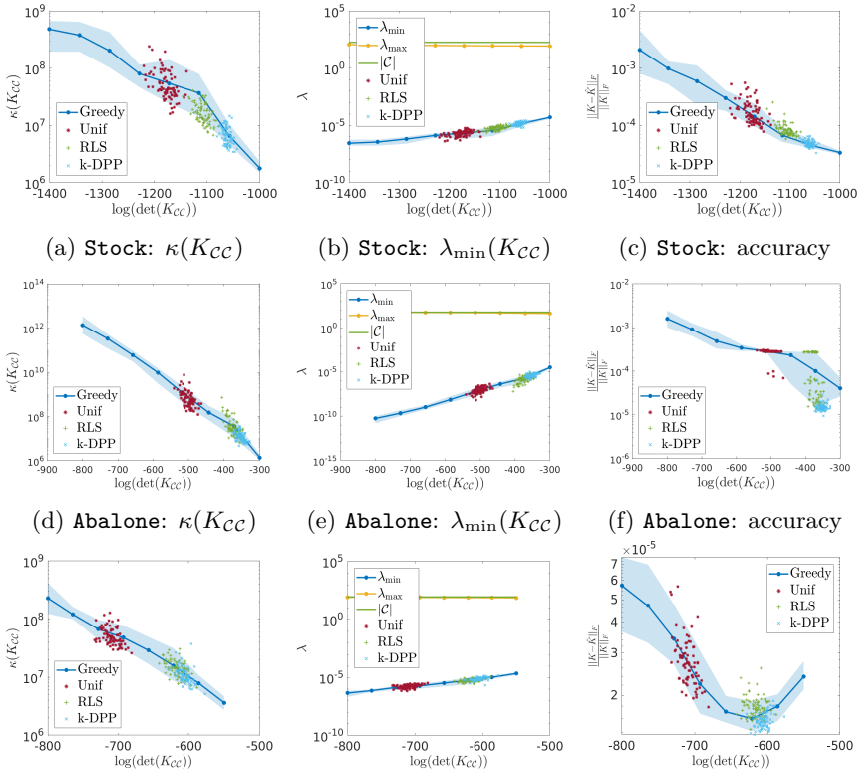
(a) `A. Credit`: $\kappa(K_{CC})$ (b) `Glass`: $\kappa(K_{CC})$ (c) `B. Cancer`: $\kappa(K_{CC})$

(d) `A. Credit`: NMI (e) `Glass`: NMI (f) `B. Cancer`: NMI

Figure A.3.2: Clustering results. The condition number and NMI are plotted as a function of $\log(\det(K_{CC}))$. A large NMI corresponds to a good accuracy.

Table A.3.2: Datasets and parameters used for the experiments.

Dataset	Task	n	d	σ	λ	$d_{\text{eff}}(n\lambda)$
`Stock`	Kernel approx.	950	10	5	10^{-6}	119
`Abalone`	Kernel approx.	4177	8	10	10^{-6}	37
`Bank 8FM`	Kernel approx.	8192	8	10	10^{-6}	95
`Parkinson`	KPCA	5875	20	10	10^{-6}	219
`Wine Quality`	KPCA	6497	11	10	10^{-6}	133
`Housing`	KRR	506	13	3	10^{-4}	110
`Parkinson`	KRR	5875	20	3	10^{-4}	343

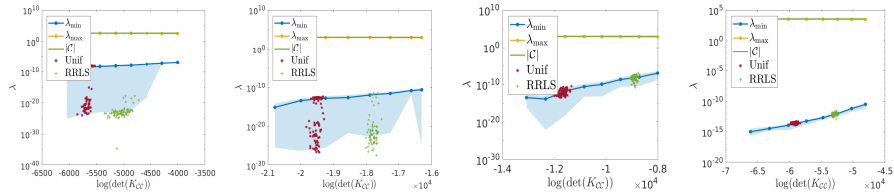


(a) `Stock`: $\kappa(K_{\mathcal{CC}})$ (b) `Stock`: $\lambda_{\min}(K_{\mathcal{CC}})$ (c) `Stock`: accuracy

(d) `Abalone`: $\kappa(K_{\mathcal{CC}})$ (e) `Abalone`: $\lambda_{\min}(K_{\mathcal{CC}})$ (f) `Abalone`: accuracy

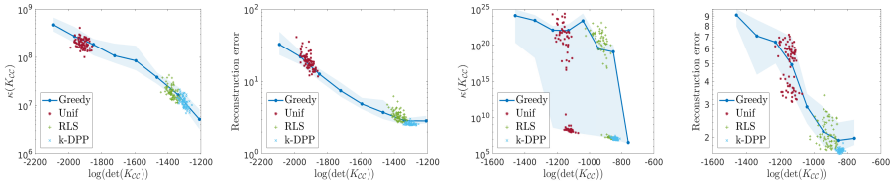
(g) `Bank 8FM`: $\kappa(K_{\mathcal{CC}})$ (h) `Bank 8FM`: $\lambda_{\min}(K_{\mathcal{CC}})$ (i) `Bank 8FM`: accuracy

Figure A.3.3: Kernel approximation results. The condition number, smallest and largest eigenvalues of $K_{\mathcal{CC}}$, relative Frobenius norm of the Nyström approximation error versus $\log(\det(K_{\mathcal{CC}}))$.



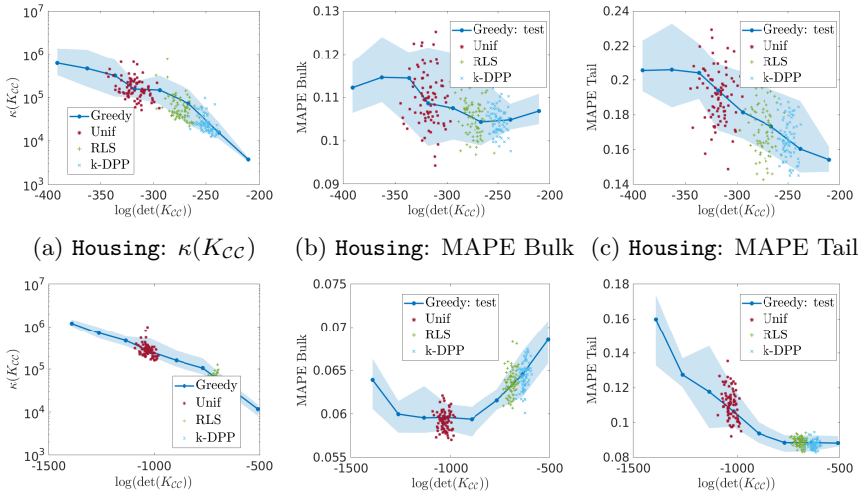
(a) `MiniBooNE` (b) `codRNA` (c) `Adult` (d) `Covertype`

Figure A.3.4: Results accompanying the large-scale Nyström and KPCA experiments. The smallest and largest eigenvalues of $K_{\mathcal{CC}}$ are plotted as a function of $\log(\det(K_{\mathcal{CC}}))$.



(a) `Park.`: $\kappa(K_{CC})$ (b) `Park.`: error (c) `Wine Q.`: $\kappa(K_{CC})$ (d) `Wine Q.`: error

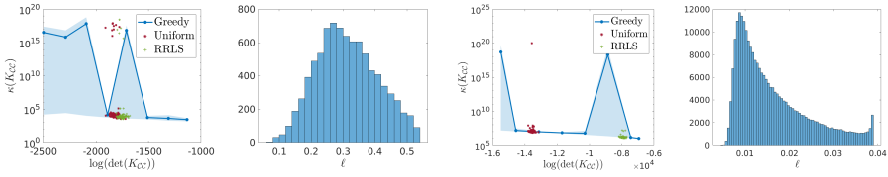
Figure A.3.5: KPCA results. The condition number and reconstruction error using half of the components are plotted as a function of $\log(\det(K_{CC}))$.



(a) `Housing`: $\kappa(K_{CC})$ (b) `Housing`: MAPE Bulk (c) `Housing`: MAPE Tail

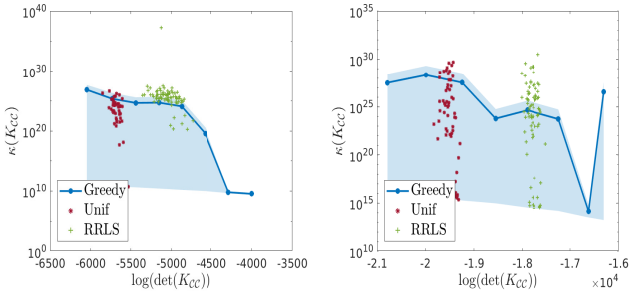
(d) `Park.`: $\kappa(K_{CC})$ (e) `Park.`: MAPE Bulk (f) `Park.`: MAPE Tail

Figure A.3.6: Regression results. The condition number and MAPE on the test set are plotted as a function of $\log(\det(K_{CC}))$.



(a) `Bike S.`: $\kappa(K_{CC})$ (b) `Bike S.`: RLS (c) `Year P.`: $\kappa(K_{CC})$ (d) `Year P.`: RLS

Figure A.3.7: Results accompanying the large-scale regression experiments. The condition number as a function of $\log(\det(K_{CC}))$ and the approximate RLS distribution are visualized.



(a) `MiniBooNE`:$\kappa(K_{CC})$ (b) `codRNA`: $\kappa(K_{CC})$

Figure A.3.8: Results accompanying the large-scale Nyström experiments. The condition number of K_{CC} versus the sample diversity.

Appendix B

Nyström landmark sampling and regularized Christoffel functions

This appendix contains the supplementary material of the work presented in Section 3.1. Section B.1 defines the Christoffel function. In Section B.2, some technical lemmata are gathered. In Sections B.3 and B.4, the proofs are given related to the DAS and RAS algorithms respectively. Section B.5 contains an ablation study for different bandwidths of the Gaussian kernel, whereas Section B.6 contains simulation details, additional tables and figures.

B.1 Christoffel function and orthogonal polynomials

For completeness, we simply recall here the definition of the Christoffel function [177]. Let $\ell \in \mathbb{N}$ and $p(x)$ be an integrable real function on $\mathbb{R}^d$. Then, the Christoffel function reads:

$$z \mapsto \min_{P \in \mathbb{R}_\ell(X)} \int P^2(x) p(x) \, \mathrm{d}\, x \text{ s.t. } P(z) = 1,$$

where $\mathbb{R}_\ell(X)$ is the set of d variate polynomials of degree at most ℓ.

In comparison, the optimization problem (CF) involves a minimization over a RKHS and includes an additional regularization term. Let $f \in \mathcal{H}$ and $\alpha \in \mathbb{R}^n$. Then, denote the sampling operator $\mathfrak{S} : \mathcal{H} \to \mathbb{R}^n$, such that $\mathfrak{S}f = \frac{1}{\sqrt{n}}(f(\boldsymbol{x}_1) \dots f(\boldsymbol{x}_n))^\top$ and its adjoint $\mathfrak{S}^* : \mathbb{R}^n \to \mathcal{H}$ given by $\mathfrak{S}^*\alpha = \frac{1}{\sqrt{n}} \sum_{i=1}^n \alpha_i k_{x_i}$. In order to rephrase the optimization problem in the RKHS, we introduce the covariance operator $\mathfrak{S}^*\mathfrak{S} = \frac{1}{n} \sum_{i=1}^n k_{x_i} \otimes k_{x_i}$ and the kernel matrix $\mathfrak{S}\mathfrak{S}^* = \frac{1}{n}K$ which share the same non-zero eigenvalues. Then, the problem (CondCF) reads:

$$C_{\mathcal{C}}(\boldsymbol{x}_z) = \inf_{f \in \mathcal{H}} \langle f, (\mathfrak{S}^*\mathfrak{S} + \gamma \mathbb{I})f \rangle_{\mathcal{H}} \text{ s.t. } \langle k_{x_z}, f \rangle_{\mathcal{H}} = 1 \text{ and } f \in (\operatorname{span}\{k_{x_s} | s \in \mathcal{C}\})^\perp. \tag{B.1.1}$$

In other words, the formulation (B.1.1) corresponds to the definition (CF) where the RKHS $\mathcal{H}$ is replaced by a specific subspace of $\mathcal{H}$.

B.2 Useful results and Deferred Proofs

Useful results

We firstly give in Lemma B.2.1 a result which is used to compute the expression of conditioned Christoffel functions in Proposition 3.1.2.

Lemma B.2.1. *Let H a Hilbert space. Let $\{\boldsymbol{v}_s \in H | s \in \mathcal{C} \subseteq [n]\}$ a collection of vectors and denote $V = \operatorname{span}\{\boldsymbol{v}_s \in H | s \in \mathcal{C} \subseteq [n]\}$. Define $\pi_{V^\perp}$ the linear projector onto the orthogonal of V. Let $\boldsymbol{u} \in H$ such that $\pi_{V^\perp}\boldsymbol{u} \neq 0$. Then, the optimal value of*

$$m^\star = \min_{x \in H} \langle \boldsymbol{x}, \boldsymbol{x} \rangle \text{ s.t. } \begin{cases} \langle \boldsymbol{x}, \boldsymbol{u} \rangle = 1 \\ \langle \boldsymbol{x}, \boldsymbol{v}_s \rangle = 0, \text{ for all } s \in \mathcal{C} \end{cases} \tag{B.2.1}$$

is given by $m^\star = 1/\|\pi_{V^\perp}\boldsymbol{u}\|^2$.

Proof. Let $\boldsymbol{x} = \boldsymbol{x}_\perp + \boldsymbol{x}_\parallel$ where $\boldsymbol{x}_\perp = \pi_{V^\perp}\boldsymbol{x}$ and $\boldsymbol{x}_\parallel = \pi_V \boldsymbol{x}$. Then, the objective satisfies

$$\langle \boldsymbol{x}, \boldsymbol{x} \rangle = \langle \boldsymbol{x}_\perp, \boldsymbol{x}_\perp \rangle + \langle \boldsymbol{x}_\parallel, \boldsymbol{x}_\parallel \rangle \geq \langle \boldsymbol{x}_\perp, \boldsymbol{x}_\perp \rangle.$$

This indicates that we can simply optimize over $\boldsymbol{x} \in V^\perp$. The problem is then

$$m^\star = \min_{\boldsymbol{x} \in V^\perp} \langle \boldsymbol{x}, \boldsymbol{x} \rangle \text{ s.t. } \langle \boldsymbol{x}, \pi_{V^\perp}\boldsymbol{u} \rangle = 1.$$

By using a Lagrange multiplier $\lambda \in \mathbb{R}$, we find $2\boldsymbol{x} + \lambda \pi_{V^\perp}\boldsymbol{u} = 0$. Next, we use the constraint $\langle \boldsymbol{x}, \pi_{V^\perp}\boldsymbol{u} \rangle = 1$. Thus, we find that the multiplier has to satisfy $\lambda = -2/\|\pi_{V^\perp}\boldsymbol{u}\|^2$. After a substitution, this yields finally $\boldsymbol{x}^\star = \pi_{V^\perp}\boldsymbol{u}/\|\pi_{V^\perp}\boldsymbol{u}\|^2$. □

Next, we recall in Lemma B.2.2 an instrumental identity which is often called 'push-through identity'.

Lemma B.2.2 (Push-through identity)**.** *Let X and Y be $n \times m$ matrices such that $X^\top Y + \mathbb{I}_m$ and $YX^\top + \mathbb{I}_n$ are invertible. Then, we have*

$$(X^\top Y + \mathbb{I}_m)^{-1} X^\top = X^\top (YX^\top + \mathbb{I}_n)^{-1}.$$

The following identity is a slightly generalized version of a result given in [165], which originally considers only the diagonal elements of the matrix $K - L_{\mu,S}(K)$ below.

Lemma B.2.3 (Lemma 6 in Supplementary Material F of [165])**.** *Let $\mu > 0$ and S be a sampling matrix. Let B such that $K = B^\top B$. We have*

$$K - L_{\mu,S}(K) = \mu B^\top (BSS^\top B^\top + \mu \mathbb{I})^{-1} B.$$

Proof. This is a consequence of the push-through identity in Lemma B.2.2. □

In what follows, we list two results in Lemma B.2.4 and Lemma B.2.5 which are used in the proof of Proposition 3.1.4.

Lemma B.2.4. *Let B be a $n \times n$ matrix. Let S be a $n \times k$ matrix and $\alpha \in \mathbb{R}$, such that $\mathbb{I} + \alpha B^\top SS^\top B$ and $\mathbb{I} + \alpha S^\top BB^\top S$ are non-singular. Then, we have*

$$B(\mathbb{I} + \alpha B^\top SS^\top B)^{-1} B^\top = BB^\top - \alpha BB^\top S(\mathbb{I} + \alpha S^\top BB^\top S)^{-1} S^\top BB^\top.$$

Proof. We consider the expression on the RHS. It holds that

$$\begin{aligned}
&BB^\top - \alpha BB^\top S(\mathbb{I} + \alpha S^\top BB^\top S)^{-1} S^\top BB^\top \\
&= B(\mathbb{I} - \alpha B^\top S(\mathbb{I} + \alpha S^\top BB^\top S)^{-1} S^\top B) B^\top \\
&= B(\mathbb{I} - \alpha B^\top SS^\top B(\mathbb{I} + \alpha B^\top SS^\top B)^{-1}) B^\top = B(\mathbb{I} + \alpha B^\top SS^\top B)^{-1} B^\top,
\end{aligned}$$

where the next to last equality is due to the push-through identity in Lemma B.2.2 with $X = \alpha S^\top B$ and $Y = S^\top B$. This yields the desired result. □

Lemma B.2.5. *Let $K \succeq 0$ and let M be $n \times n$ symmetric matrix such that $K + M \succ 0$. Define $P_M = K^{1/2} (K + M)^{-1} K^{1/2}$. Let S be a $n \times k$ matrix and $\alpha \in \mathbb{R}$ be such that $K + M + \alpha K^{1/2} SS^\top K^{1/2}$ and $\mathbb{I} + \alpha S^\top P_M S$ are non-singular. Then, we have*

$$K^{1/2} \left(K + M + \alpha K^{1/2} SS^\top K^{1/2}\right)^{-1} K^{1/2} = P_M - \alpha P_M S \left(\mathbb{I} + \alpha S^\top P_M S\right)^{-1} S^\top P_M.$$

Proof. The results follows from Lemma B.2.4 by choosing $B = K^{1/2}(K+M)^{-1/2}$ such that $BB^\top = P_M$. □

Proof of Proposition 3.1.2

Proof. We start by setting up some notations. Let $f \in \mathcal{H}$ and $\alpha \in \mathbb{R}^n$. Then, define the sampling operator $\mathfrak{S} : \mathcal{H} \to \mathbb{R}^n$, such that $\mathfrak{S}f = \frac{1}{\sqrt{n}}(f(\boldsymbol{x}_1)\dots f(\boldsymbol{x}_n))^\top$. Its adjoint writes $\mathfrak{S}^* : \mathbb{R}^n \to \mathcal{H}$ and is given by $\mathfrak{S}^*\alpha = \frac{1}{\sqrt{n}}\sum_{i=1}^n \alpha_i k_{x_i}$.

Firstly, since $z \in X$, the objective (CondCF) and the constraints depend only on the discrete set $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$. Then, we can decompose $f = f_X + f_X^\perp$ where $\langle f_X, f_X^\perp\rangle_\mathcal{H} = 0$ and $f_X \in \operatorname{span}\{k_{x_i} | i \in [n]\}$. Since, by construction, $f_X^\perp(\boldsymbol{x}_i) = 0$ for all $i \in [n]$, the objective satisfies:

$$\frac{1}{n}\sum_{i=1}^n f_X(\boldsymbol{x}_i)^2 + \gamma\|f\|_\mathcal{H}^2 \geq \frac{1}{n}\sum_{i=1}^n f_X(\boldsymbol{x}_i)^2 + \gamma\|f_X\|_\mathcal{H}^2.$$

Hence, we can rather minimize on $f \in \operatorname{span}\{k_{x_i} | i \in [n]\}$. This means that we can find $\alpha \in \mathbb{R}^n$ such that $f^\star = \sum_{i=1}^n \alpha_i k_{x_i} = \sqrt{n}\mathfrak{S}^*\alpha$. The objective (CondCF) reads

$$\begin{aligned}
\langle f^\star, (\mathfrak{S}^*\mathfrak{S} + \gamma\mathbb{I})f^\star\rangle_\mathcal{H} &= \sqrt{n}\alpha^\top\mathfrak{S}(\mathfrak{S}^*\mathfrak{S} + \gamma\mathbb{I})\mathfrak{S}^*\alpha\sqrt{n} \\
&= n\alpha^\top(\mathfrak{S}\mathfrak{S}^* + \gamma\mathbb{I})\mathfrak{S}\mathfrak{S}^*\alpha \\
&= n\alpha^\top(K/n + \gamma\mathbb{I})(K/n)\alpha \\
&= n^{-1}\alpha^\top(K + n\gamma\mathbb{I})K\alpha.
\end{aligned}$$

For simplicity, denote by $K_{x_i} = K\boldsymbol{e}_i$ a column of the matrix K, where $\boldsymbol{e}_i$ is the i-th element of the canonical basis of $\mathbb{R}^n$. The problem becomes then:

$$\min_{\alpha\in\mathbb{R}^n} \alpha^\top M\alpha \text{ such that } \alpha^\top K_{x_z} = 1 \text{ and } \alpha^\top K_{x_s} = 0 \text{ for all } s \in \mathcal{C},$$

where $M = n^{-1}K(K + n\gamma\mathbb{I})$. Recall that, since we assumed $k \succ 0$, then K is invertible. By doing the change of variables $\alpha_M = M^{1/2}\alpha$, we obtain:

$$\min_{\alpha_M\in\mathbb{R}^n} \alpha_M^\top\alpha_M \text{ such that } \alpha_M^\top K_{x_z}^{(M)} = 1 \text{ and } \alpha_M^\top K_{x_s}^{(M)} = 0 \text{ for all } s \in \mathcal{C},$$

with $K_{x_z}^{(M)} = M^{-1/2}K_{x_z}$ and $K_{x_s}^{(M)} = M^{-1/2}K_{x_s}$. Let $V = \operatorname{span}\{K_{x_s}^{(M)} | s \in \mathcal{C}\}$ and $K^{(M)} \in \mathbb{R}^{n\times|\mathcal{C}|}$ be the matrix whose columns are the vectors $K_{x_s}^{(M)}$. Notice that the elements of $\{K_{x_s}^{(M)} | s \in \mathcal{C}\}$ are linearly independent since $K_{x_s}^{(M)} = M^{-1/2}Ke_s$. Then, we use Lemma B.2.1, and find $C_\mathcal{C}^{-1}(\boldsymbol{x}_z) = \|\pi_{V^\perp}K_{x_z}^{(M)}\|^2$. By using $\pi_V = K^{(M)}G^{-1}K^{(M)\top}$ with the Gram matrix $G = K^{(M)\top}K^{(M)}$, we find:

$$\|\pi_{V^\perp}K_{x_z}^{(M)}\|^2 = K_{x_z}^{(M)\top}\left(\mathbb{I} - K^{(M)}G^{-1}K^{(M)\top}\right)K_{x_z}^{(M)}. \tag{B.2.2}$$

By substituting $M = n^{-1}K(K+n\gamma\mathbb{I})$, we find that $G_{ss'} = ne_s^\top K(K+n\gamma\mathbb{I})^{-1}e_{s'}$ for all $s, s' \in \mathcal{C}$. Hence, we find:

$$\|\pi_{V^\perp} K_{x_z}^{(M)}\|^2 = n\left\{e_z^\top K(K+n\gamma\mathbb{I})^{-1}e_z - e_z^\top K(K+n\gamma\mathbb{I})^{-1}G^{-1}K(K+n\gamma\mathbb{I})^{-1}e_z\right\},$$

which yields the desired result. □

Details of derivations of Section 3.1.3

By using a similar approach and the same notations as in the previous subsection, we find

$$C_{\boldsymbol{\eta}}(\boldsymbol{x}_z) = \min_{\alpha\in\mathbb{R}^n} \alpha^\top \left(\frac{1}{n}K\,\mathrm{Diag}(\boldsymbol{\eta})K + \gamma K\right)\alpha \text{ such that } \alpha^\top K_{x_z} = 1,$$

for some $z \in [n]$. By using Lemma B.2.1 we find the expression given in (3.1.5)

$$\begin{aligned} C_{\boldsymbol{\eta}}(\boldsymbol{x}_z) &= \left(K_{x_z}^\top \left(\frac{1}{n}K\,\mathrm{Diag}(\boldsymbol{\eta})K + \gamma K\right)^{-1} K_{x_z}\right)^{-1} \\ &= \frac{n^{-1}}{\left[K\left(K\,\mathrm{Diag}(\boldsymbol{\eta})K + n\gamma K\right)^{-1}K\right]_{zz}}. \end{aligned}$$

The second expression for Christoffel functions given in (3.1.6) is merely obtained by using the matrix inversion lemma as follows

$$\begin{aligned} K\left(K\,\mathrm{Diag}(\boldsymbol{\eta})K + n\gamma K\right)^{-1}K &= K\left(\frac{K^{-1}}{n\gamma} - \left(\mathrm{Diag}(\boldsymbol{\eta})^{-1} + \frac{K}{n\gamma}\right)^{-1}\right)K \\ &= \frac{1}{n\gamma}\left(K - K\left(K + n\gamma\,\mathrm{Diag}(\boldsymbol{\eta})\right)^{-1}\right) \\ &= K\left(K + n\gamma\,\mathrm{Diag}(\boldsymbol{\eta})^{-1}\right)^{-1}\mathrm{Diag}(\boldsymbol{\eta})^{-1}. \end{aligned}$$

By taking the (z, z)-th entry of the matrix above gives directly (3.1.6).

Proof of Proposition 3.1.3

Proof. The objective $C_{\mathcal{C}}(\boldsymbol{x}_z)^{-1}$ can be simply obtained by calculating the following determinant

$$n^{-1}C_{\mathcal{C}}(\boldsymbol{x}_z)^{-1} = \det\left(\begin{pmatrix} P_{zz} & P_{z,\mathcal{C}} \\ P_{z,\mathcal{C}}^\top & P_{\mathcal{C},\mathcal{C}} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0^\top & P_{\mathcal{C},\mathcal{C}} \end{pmatrix}^{-1}\right)$$

$$= \det\begin{pmatrix} P_{zz} & P_{z,\mathcal{C}}P_{\mathcal{C},\mathcal{C}}^{-1} \\ P_{z,\mathcal{C}}^\top & \mathbb{I} \end{pmatrix} = P_{zz} - P_{z,\mathcal{C}}P_{\mathcal{C},\mathcal{C}}^{-1}P_{z,\mathcal{C}}^\top,$$

where we use the well-known formula for the determinant of block matrices at the last equality. This yields the desired result. □

Proof of Proposition 3.1.4

Proof. By using Lemma B.2.1 or the results of [177], we find the expression

$$n^{-1}(\boldsymbol{e}_i^\top \underbrace{K(K\,\mathrm{Diag}(\boldsymbol{\eta})K + n\gamma K)^{-1}K}_{T}\boldsymbol{e}_i)^{-1} = \inf_{f(\boldsymbol{x}_z)=1} \frac{1}{n}\sum_{i=1}^{n} \eta_i f(\boldsymbol{x}_i)^2 + \gamma\|f\|_{\mathcal{H}}^2, \tag{B.2.3}$$

where the weights are defined as $\eta_i = 1 + \mu^{-1}$ for all $i \in \mathcal{C}$ and $\eta_i = \nu$ if $i \in \tilde{\mathcal{C}}$ and $\eta_i = 1$ otherwise. Recall that $\mu > 0$ and $0 < \nu < 1$ so that $\mathrm{Diag}(\boldsymbol{\eta})$ is the positive diagonal matrix

$$\mathrm{Diag}(\boldsymbol{\eta}) = \mathbb{I} + \mu^{-1}CC^\top + (\nu - 1)\tilde{C}\tilde{C}^\top,$$

where C (resp. $\tilde{C}$) is a sampling matrix obtained by sampling the columns of the identity matrix belonging to $\mathcal{C}$ (resp. $\tilde{\mathcal{C}}$). Notice that $CC^\top$ (resp. $\tilde{C}\tilde{C}^\top$) is the diagonal matrices with entries equal to 1 for elements indexed by $\mathcal{C}$ (resp. $\tilde{\mathcal{C}}$) and equal to zero otherwise. First, we establish an equivalent expression for T at the LHS of (B.2.3), as follows

$$T = K(K\,\mathrm{Diag}(\boldsymbol{\eta})K + n\gamma K)^{-1}K$$

$$= K^{1/2}\left(K^{1/2}\left(\mathbb{I} + \mu^{-1}CC^\top + (\nu-1)\tilde{C}\tilde{C}^\top\right)K^{1/2} + n\gamma\mathbb{I}\right)^{-1}K^{1/2}.$$

Then, the idea is to define the symmetric matrix $M = (\nu-1)K^{1/2}\tilde{C}\tilde{C}^\top K^{1/2} + n\gamma\mathbb{I}$, which is such that

$$K + M = K^{1/2}(\mathbb{I} - \tilde{C}\tilde{C}^\top)K^{1/2} + \nu K^{1/2}\tilde{C}\tilde{C}^\top K^{1/2} + n\gamma\mathbb{I} \succ 0, \tag{B.2.4}$$

since $\mathbb{I} - \tilde{C}\tilde{C}^\top \succeq 0$. Then, define the following notation (also used in Lemma B.2.5):

$$P_M = K^{1/2}(K+M)^{-1}K^{1/2} = K^{1/2}\left(K + n\gamma\mathbb{I} - (1-\nu)K^{1/2}\tilde{C}\tilde{C}^\top K^{1/2}\right)^{-1}K^{1/2},$$

which satisfies $P_M \succ 0$ since $K+M \succ 0$ and $K \succ 0$. Next, by using Lemma B.2.5 with $\alpha = \mu^{-1}$ and the matrix M defined above (B.2.4), we find the following equivalent expression for T defined in (B.2.3), i.e.,

$$\begin{aligned} T &= K^{1/2}\left(K + \underbrace{(\nu-1)K^{1/2}\tilde{C}\tilde{C}^\top K^{1/2} + n\gamma\mathbb{I}}_{M} + \mu^{-1}K^{1/2}CC^\top K^{1/2}\right)^{-1}K^{1/2} \\ &= P_M - \mu^{-1}P_M C\left(\mathbb{I} + \mu^{-1}C^\top P_M C\right)^{-1}C^\top P_M. \end{aligned}$$

Now, in order to match the notation of Proposition 3.1.4, we define $H = P_M$. This gives

$$T = H - \mu^{-1}HC\left(\mathbb{I} + \mu^{-1}C^\top HC\right)^{-1}C^\top H = H - L_{\mu,C}(H). \tag{B.2.5}$$

Next, to find an equivalent expression for

$$H = K^{1/2}\left(K + n\gamma\mathbb{I} + (\nu-1)K^{1/2}\tilde{C}\tilde{C}^\top K^{1/2}\right)^{-1}K^{1/2},$$

we use once more Lemma B.2.5, but this time we take $\alpha = \nu - 1$ and $M = n\gamma\mathbb{I}$. Notice that α does not need to be positive to be able to apply Lemma B.2.5. An equivalent expression for H is

$$H = P + (1-\nu)P\tilde{C}\left(\mathbb{I} - (1-\nu)\tilde{C}^\top P\tilde{C}\right)^{-1}\tilde{C}^\top P, \tag{B.2.6}$$

with $P = K(K+n\gamma\mathbb{I})^{-1}$. Remark that the assumptions of Lemma B.2.5 are satisfied: on the one hand, the matrix inverse in (B.2.6) exits since $0 \preceq \tilde{C}^\top P\tilde{C} \preceq \mathbb{I}$ and $0 < \nu < 1$; on the other hand, the matrix $K + n\gamma\mathbb{I} + (\nu-1)K^{1/2}\tilde{C}\tilde{C}^\top K^{1/2}$ is non-singular in the light of (B.2.4). By combining (B.2.5) and (B.2.6), we have the desired result. □

B.3 Proof of the results of DAS

Proof of Proposition 3.1.5

Proof. Let us explain how Algorithm 3.1.1 can be rephrased by considering first Proposition 3.1.3. First, we recall the spectral factorization of the projector

kernel matrix $P = B^\top B$. For simplicity, denote the Hilbert space[1] $H = \mathbb{R}^n$. We also introduce the notation $\mathcal{F} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ for the set of columns of $B \in \mathbb{R}^{n\times n}$ which is considered as a compact subset of H. The vector subspace generated by the columns indexed by the subset $\mathcal{C}_m$ is denoted by $V_m = \text{span}\{\boldsymbol{b}_s | s \in \mathcal{C}_m\} \subset H$. With these notations and thanks to Proposition 3.1.3, the *square root* of the objective maximized by Algorithm 3.1.1 can be written $\sigma_m = n^{-1/2} C_{\mathcal{C}_m}^{-1/2}(\boldsymbol{x}_{s_{m+1}}) = \|\boldsymbol{b}_{s_{m+1}} - \pi_{V_m}\boldsymbol{b}_{s_{m+1}}\|_2$. Hence, by using the definition of the Christoffel function, we easily see that $\sigma_{m+1} \leq \sigma_m$. Then, the algorithm can be viewed as a greedy reduced subspace method, namely:

$$\sigma_m = \max_{b\in\mathcal{F}} \text{dist}(\boldsymbol{b}, V_m),$$

where the distance between a column $\boldsymbol{b} \in \mathcal{F}$ and the subspace V_m is given by:

$$\text{dist}(b, V_m) = \min_{X\in V_m} \|\boldsymbol{b} - X\|_2. \tag{B.3.1}$$

As explained in [61, 80, 21], the convergence of this type of algorithm can be studied as a function of the Kolmogorov width:

$$d_m = \min_{Y_m} \max_{\boldsymbol{b}\in\mathcal{F}} \text{dist}(\boldsymbol{b}, Y_m), \tag{B.3.2}$$

where the minimization is over all m-dimensional vector subspace Y_m of H. Intuitively, d_m is the best approximation error calculated over all possible subspace of dimension m with $m < n$. Then, Theorem 3.2 together with Corollary 3.3 in [61] gives the upper bound:

$$\sigma_{2m} \leq \sqrt{2\|P\|_\infty d_m}, \quad \text{for all } m \geq 1, \tag{B.3.3}$$

which is a close analogue of eq. (4.9) in [80]. Hence, by upper bounding the Kolmogorov width, we provide a way to calculate a convergence rate of the greedy algorithm. To do so, we first recall that $\mathcal{F} = \{\boldsymbol{b}_1, \ldots \boldsymbol{b}_n\}$ and where the i-th column of B is denoted by $\boldsymbol{b}_i = B\boldsymbol{e}_i$. Therefore, we have:

$$\max_{\boldsymbol{b}\in\mathcal{F}} \min_{\boldsymbol{y}\in Y_m} \|\boldsymbol{b}-\boldsymbol{y}\|_2 = \max_{i\in[N]} \min_{\boldsymbol{y}\in Y_m} \|B\boldsymbol{e}_i - \boldsymbol{y}\|_2 \leq \max_{\|\boldsymbol{x}\|_2\leq 1} \min_{\boldsymbol{y}\in Y_m} \|B\boldsymbol{x} - \boldsymbol{y}\|_2, \tag{B.3.4}$$

where the inequality is obtained by noticing that $\boldsymbol{e}_i$ is such that $\|\boldsymbol{e}_i\|_2 = 1$ and by observing that maximizing an objective over a larger set yields a larger objective value. Then, in the light of this remark and according to [181], we define a modified width as follows:

$$\tilde{d}_m(B) = \min_{Y_m} \max_{\|\boldsymbol{x}\|_2\leq 1} \min_{\boldsymbol{y}\in Y_m} \|B\boldsymbol{x} - \boldsymbol{y}\|_2,$$

[1] Here, we do not endow H with a RKHS structure.

which naturally satisfies $d_m \leq \tilde{d}_m(B)$ as it can be seen by taking the minimum of both sides of (B.3.4) over the m-dimensional vector spaces Y_m and by recalling the Kolmogorov width definition in (B.3.2). Then, $\tilde{d}_m(B)$ can be usefully upper bounded thanks to Theorem B.3.1 that we adapt from [181].

Theorem B.3.1 (Thm 1.1 in [181])**.** *Let B be a real $p \times q$ matrix. Let $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$ denote the p eigenvalues of $BB^\top$ and let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ denote a corresponding set of orthonormal eigenvectors. Then, $\tilde{d}_m(B) = \lambda_{m+1}^{1/2}$ if $m < p$ and $\tilde{d}_m(B) = 0$ if $m \geq p$. Furthermore if $m < p$, then an optimal subspace for $\tilde{d}_m$ is* $\operatorname{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$.

Our convergence result given in Corollary 3.1.5 simply follows from (B.3.3), the inequality $d_m \leq \tilde{d}_m$ and Theorem B.3.1 by noting that the non-zero eigenvalues of $BB^\top$ are the non-zero eigenvalues of $B^\top B$. Notice that B is obtained by the spectral factorization of $P = B^\top B$ and therefore B is full rank. □

Proof of Lemma 3.1.1

Proof. To prove this result, we rely on Lemma B.2.3 and on the definition of $P = K(K + n\gamma\mathbb{I})^{-1}$. Firstly, by following Lemma B.2.3, we have

$$K - L_{\mu,S}(K) = \mu K^{1/2}(K^{1/2}SS^\top K^{1/2} + \mu\mathbb{I})^{-1}K^{1/2}, \tag{B.3.5}$$

with $\mu > 0$. Then, we can obtain a similar expression for the projector kernel

$$\begin{aligned} P - L_{\mu,S}(P) &= \mu P^{1/2}(P^{1/2}SS^\top P^{1/2} + \mu\mathbb{I})^{-1}P^{1/2} \\ &= \mu K^{1/2}\left(K^{1/2}SS^\top K^{1/2} + \mu(K + n\gamma\mathbb{I})\right)^{-1}K^{1/2}, \end{aligned}$$

where we substituted the definition $P^{1/2} = K^{1/2}(K + n\gamma\mathbb{I})^{-1/2}$ in the last equality. Next, we use $K \preceq \lambda_{max}(K)\mathbb{I}$, and the property that $A^{-1} \succeq B^{-1}$ if $B \succeq A$ with A and $B \succ 0$. We obtain:

$$\left(K^{1/2}SS^\top K^{1/2} + \mu(K + n\gamma\mathbb{I})\right)^{-1} \succeq \left(K^{1/2}SS^\top K^{1/2} + \mu(\lambda_{max}(K) + n\gamma)\mathbb{I}\right)^{-1},$$

and the result follows from the following inequality

$$\begin{aligned} P - L_{\mu,S}(P) &\succeq \mu K^{1/2}\left(K^{1/2}SS^\top K^{1/2} + \mu(\lambda_{max}(K) + n\gamma)\mathbb{I}\right)^{-1}K^{1/2} \\ &= (\lambda_{max}(K) + n\gamma)^{-1}\tilde{\mu}K^{1/2}\left(K^{1/2}SS^\top K^{1/2} + \tilde{\mu}\mathbb{I}\right)^{-1}K^{1/2} \\ &= (\lambda_{max}(K) + n\gamma)^{-1}\left(K - L_{\tilde{\mu},S}(K)\right), \end{aligned}$$

where we used again (B.3.5) in the last equality and defined the following $\tilde{\mu} = \mu(\lambda_{max}(K) + n\gamma)$. □

B.4 Proof of the results of RAS

Proof of Lemma 3.1.3

The statement of Lemma 3.1.3 has been adapted from [73] to the setting of Section 3.1. Its proof, given for completeness, follows the same lines as in Lemma 1 in the Supplementary Material of [73].

Proof. For simplicity let $\theta = \epsilon n\gamma/(1+\epsilon)$. By assumption, we have

$$\Psi\Psi^\top - \Psi SS^\top \Psi^\top \preceq t\mathbb{I},$$

with $P_\epsilon(P_{n\gamma}(K)) = \Psi^\top\Psi$. By using the composition rule given in Lemma 3.1.2, we find the equivalent expression

$$\Psi^\top\Psi = P_\epsilon(P_{n\gamma}(K)) = \frac{1}{1+\epsilon}P_\theta(K) \text{ with } \theta = \frac{\epsilon n\gamma}{1+\epsilon}.$$

Therefore, we can choose the factorization given by $\Psi = (1+\epsilon)^{-1/2}(K + \theta\mathbb{I})^{-1/2}K^{1/2}$. By using our starting hypothesis and by denoting for simplicity $P_\theta = P_\theta(K)$, we have

$$\frac{1}{1+\epsilon}\left(P_\theta - P_\theta^{1/2}SS^\top P_\theta^{1/2}\right) \preceq t\mathbb{I}.$$

After a substitution of $P_\theta(K) = K(K+\theta\mathbb{I})^{-1}$, this gives the following inequality

$$(K+\theta\mathbb{I})^{-1/2}\left(K - K^{1/2}SS^\top K^{1/2}\right)(K+\theta\mathbb{I})^{-1/2} \preceq t(1+\epsilon)\mathbb{I}.$$

Equivalently, we have the following inequality

$$K - K^{1/2}SS^\top K^{1/2} \preceq t(1+\epsilon)(K+\theta\mathbb{I}),$$

which gives

$$K^{1/2}SS^\top K^{1/2} \succeq K - t(1+\epsilon)(K+\theta\mathbb{I}). \tag{B.4.1}$$

Then, if $1 - t(1+\epsilon) > 0$, it holds that

$$\begin{aligned} K - L_{\theta,S}(K) &= \theta K^{1/2}\big(K^{1/2}SS^\top K^{1/2} + \theta\mathbb{I}\big)^{-1}K^{1/2} \quad \text{(by Lemma B.2.3)} \\ &\preceq \theta K^{1/2}\big(K + \theta\mathbb{I} - t(1+\epsilon)(K+\theta\mathbb{I})\big)^{-1}K^{1/2} \quad \text{(by (B.4.1))} \\ &= \frac{\theta}{1-t(1+\epsilon)}P_\theta(K), \end{aligned}$$

where we used $P_\theta(K) = K(K + \theta\mathbb{I})^{-1}$ at the last equality. Now, by using Lemma 3.1.2, we find

$$K - L_{\frac{\epsilon n\gamma}{1+\epsilon},S}(K) \preceq \frac{\epsilon n\gamma/(1+\epsilon)}{1 - t(1+\epsilon)}(1+\epsilon)P_\epsilon(P_{n\gamma}(K)),$$

which is the stated result. □

Proof of Theorem 3.1.1

The proof structure follows closely [47] with suitable adaptations. In particular and in contrast with [47], a main ingredient of the proof of Theorem 3.1.1 is the following improved Freedman inequality.

Theorem B.4.1 (Thm 3.2 in [161])**.** *Let $\boldsymbol{x}_1, \ldots, X_n$ be martingale differences valued in the $n \times n$ symmetric matrices and let $U > 0$ such that $\|X_i\|_2 \leq U$ almost surely. Denote the variance $W_n = \sum_{i=1}^n \mathbb{E}_{i-1}(\boldsymbol{x}_i^2)$. Then, for any $t \geq \frac{1}{6}(U + \sqrt{U^2 + 36\sigma^2})$, we have:*

$$\Pr\left(\|\sum_{i=1}^n X_i\|_2 > t, \lambda_{max}(W_n) \leq \sigma^2\right) \leq 50\,\mathrm{Tr}\left(\min\{\tfrac{t}{U}\tfrac{\mathbb{E}(W_n)}{\sigma^2}, 1\}\right)\exp\left(\tfrac{-t^2/2}{\sigma^2 + tU/3}\right). \tag{B.4.2}$$

Notice that the function $\min\{M, 1\}$, defined for all symmetric $M \succeq 0$, acts on the eigenvalues of M. In Theorem B.4.1, σ^2 denotes an upper bound on the predictable quadratic variation and not the kernel bandwidth parameter.

We now give the proof of Theorem 3.1.1.

Proof. Let $t > 0$ and $0 < \epsilon < 1$. We consider here the sampling procedure given by RAS; see Algorithm 3.1.2. We recall that we defined the matrix B such that $B^\top B = P_{n\gamma}(K)$. Then, this proof considers the problem of sampling columns of the matrix Ψ which is defined such that

$$\Psi = (BB^\top + \epsilon\mathbb{I})^{-1/2}B.$$

Proof strategy. The key idea of this proof is to define a matrix martingale $Y_0, \ldots, Y_n$ involving a sum of rank 1 matrices built from the columns of Ψ so that, at step n, we have

$$Y_n = \Psi S_n S_n^\top \Psi^\top - \Psi\Psi^\top,$$

where S_n is the sampling matrix at step n. If RAS samples enough landmarks, i.e., if S_n samples enough columns of Ψ, then Y_n will be small with high

probability. In other words, we want to show that for large enough $c > 0$, we have

$$\|Y_n\|_2 \leq t,$$

and therefore $\lambda_{\max}(Y_n) \leq t$ with a large probability. Then, by using Lemma 3.1.3 and by choosing $t = 1/2$, we can obtain an error bound on a regularized Nyström approximation

$$0 \preceq K - L_{\frac{\epsilon n\gamma}{1+\epsilon},S}(K) \preceq \frac{2\epsilon n\gamma}{1-\epsilon}\mathbb{I},$$

which is the result that we want to prove.

The proof goes as follows. The matrix martingale is firstly defined. In order to use the martingale concentration inequality, we upper bound the norm of the difference sequence which boils down to find a lower bound on the sampling probability. Finally, we use the concentration inequality find a lower bound on c so that $\|Y_n\|_2 \leq t$ with a large probability.

Defining a matrix martingale. Let S_i be the sampling matrix at step i in Algorithm 3.1.2 and p_i be the sampling probability. Define now a matrix martingale $Y_0, Y_1, \ldots, Y_n \in \mathbb{R}^{n\times n}$ such that $Y_0 = 0$ and with difference sequence $X_1, \ldots, X_n$ given by:

$$X_i = \begin{cases} (\frac{1}{p_i} - 1)\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top \text{ if column } b_i \text{ is sampled,} \\ -\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top \text{ otherwise,} \end{cases} \tag{B.4.3}$$

where p_i is given in (3.1.11) and where $\boldsymbol{\psi}_i$ is the i-th column of Ψ. It remains to show that we indeed defined a martingale. To do so, we define $\mathbb{E}_{i-1}$ as the expectation conditioned on the previous steps from 0 to $i-1$ included. We simply have $\mathbb{E}_{i-1}(\boldsymbol{x}_i) = (p_i(\frac{1}{p_i} - 1) - (1 - p_i))\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top = 0$ by a direct calculation.

Next, denote by $B_{[i-1]}$ the rectangular matrix obtained by selecting the $i-1$ first columns of B, so that $B_{[n]} = B$. So, the martingale value at step $i-1$ reads

$$Y_{i-1} = (BB^\top + \epsilon\mathbb{I})^{-1/2}\left(BS_{i-1}S_{i-1}^\top B^\top - B_{[i-1]}B_{[i-1]}^\top\right)(BB^\top + \epsilon\mathbb{I})^{-1/2}.$$

We first observe that, if $\|Y_{i-1}\|_2 < t$ at step $i-1$, then by construction $\|Y_j\|_2 < t$ for all previous steps $j < i-1$. If, at some point $\|Y_{i-1}\|_2 > t$, then we set $\boldsymbol{x}_j = 0$ for all subsequent steps $j \geq i$. By abusing notations, we denote this frozen process by the same symbol $(Y_i)_i$.

Again, if $\|Y_{i-1}\|_2 < t$, then we have

$$(BB^\top + \epsilon\mathbb{I})^{-1/2}\left(BS_{i-1}S_{i-1}^\top B^\top - B_{[i-1]}B_{[i-1]}^\top\right)(BB^\top + \epsilon\mathbb{I})^{-1/2} \prec t\mathbb{I},$$

which gives the following inequality

$$BS_{i-1}S_{i-1}^{\top}B^{\top} \prec B_{[i-1]}B_{[i-1]}^{\top} + t(BB^{\top} + \epsilon\mathbb{I}). \tag{B.4.4}$$

Notice that, by definition, B contains the columns of $B_{[i-1]}$, and therefore, the following inequality holds

$$B_{[i-1]}B_{[i-1]}^{\top} \preceq BB^{\top}. \tag{B.4.5}$$

Lower bounding the sampling probability. At this point, we want to show that the norm of the difference sequence $\boldsymbol{x}_i$ is bounded in order to use the matrix martingale concentration result which is given in Theorem B.4.1. We have the elementary upper bound on the difference sequence

$$\|X_i\|_2 \leq \max\{1, 1/p_i - 1\}\|\boldsymbol{\psi}_i\boldsymbol{\psi}_i^{\top}\|_2 \leq \frac{1}{p_i}\boldsymbol{\psi}_i^{\top}\boldsymbol{\psi}_i.$$

However, the RHS of the above bound can be arbitrary large if p_i goes to zero. By using (B.4.4) and (B.4.5), we show that the sampling probability p_i is lower bounded as follows

$$p_i \geq \min\{c\boldsymbol{\psi}_i^{\top}\boldsymbol{\psi}_i, 1\}. \tag{B.4.6}$$

The argument to prove (B.4.6) goes as follows. First, we recall the definition of the sampling probability at step i, as it is given in Section 3.1.5 by

$$p_i \triangleq \min\{1, c\tilde{l}_i\},$$

with $\tilde{l}_i \triangleq \min\{1, (1+t)s_i\}$ and

$$s_i \triangleq b_i^{\top}(BS_{i-1}S_{i-1}^{\top}B^{\top} + \epsilon\mathbb{I})^{-1}b_i = \frac{1}{\epsilon}\Big[P_{n\gamma}(K) - L_{\epsilon,S_{i-1}}(P_{n\gamma}(K))\Big]_{ii},$$

while b_i is the i-th column of B. Next, we obtain the following set of inequalities

$$\begin{aligned}
\tilde{l}_i &= \min\{1, (1+t)\boldsymbol{b}_i^{\top}(BS_{i-1}S_{i-1}^{\top}B^{\top} + \epsilon\mathbb{I})^{-1}\boldsymbol{b}_i\} \\
&\geq \min\{1, (1+t)\boldsymbol{b}_i^{\top}\Big(B_{[i-1]}B_{[i-1]}^{\top} + \epsilon\mathbb{I} + t(BB^{\top} + \epsilon\mathbb{I})\Big)^{-1}\boldsymbol{b}_i\} \quad \text{(by (B.4.4))} \\
&\geq \min\{1, (1+t)\boldsymbol{b}_i^{\top}\Big(BB^{\top} + \epsilon\mathbb{I} + t(BB^{\top} + \epsilon\mathbb{I})\Big)^{-1}\boldsymbol{b}_i\} \quad \text{(by (B.4.5))} \\
&\geq \min\{1, \boldsymbol{b}_i^{\top}(BB^{\top} + \epsilon\mathbb{I})^{-1}\boldsymbol{b}_i\} = \boldsymbol{b}_i^{\top}(BB^{\top} + \epsilon\mathbb{I})^{-1}\boldsymbol{b}_i = \boldsymbol{\psi}_i^{\top}\boldsymbol{\psi}_i.
\end{aligned}$$

So, we have $p_i \geq \min\{c\boldsymbol{\psi}_i^{\top}\boldsymbol{\psi}_i, 1\}$. If $p_i = 1$, then $\boldsymbol{x}_i = 0$ as a direct consequence of the definition in (B.4.3), so that the step i does not change the value of the martingale. Otherwise, if $p_i < 1$, we have $p_i \geq c\boldsymbol{\psi}_i^{\top}\boldsymbol{\psi}_i$.

Concentration bound. We can now estimate the probability of failure by using a concentration bound given in Theorem B.4.1. Let us check that the hypotheses of Theorem B.4.1 are satisfied. Clearly, each increment is bounded as follows:

$$\|X_i\|_2 \leq \max\{1, 1/p_i - 1\}\|\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top\|_2 \leq \frac{1}{p_i}\boldsymbol{\psi}_i^\top\boldsymbol{\psi}_i \leq 1/c \triangleq U.$$

almost surely. Similarly, by using once more $\boldsymbol{\psi}_i^\top\boldsymbol{\psi}_i \leq p_i/c$, we find:

$$\mathbb{E}_{i-1}\left(\boldsymbol{x}_i^2\right) \preceq p_i(1/p_i - 1)^2\left(\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top\right)^2 + (1-p_i)\left(\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top\right)^2 \preceq \tfrac{1}{p_i}\left(\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top\right)^2 \preceq \boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top/c.$$

Then, the predictable quadratic variation satisfies the following upper bound

$$W_n \preceq \frac{1}{c}\sum_{i=1}^{n}\boldsymbol{\psi}_i\boldsymbol{\psi}_i^\top = \frac{1}{c}P_\epsilon(P_{n\gamma}(K)) = \frac{1}{c}\frac{1}{1+\epsilon}P_{\frac{\epsilon n\gamma}{1+\epsilon}}(K) \text{ almost surely,}$$

where the last equality is due to Lemma 3.1.2. Furthermore, we have $\lambda_{max}(W_n) \leq 1/c = \sigma^2$ almost surely[2]. Now, for a symmetric matrix M, we recall that $\min\{M, 1\}$ is defined on the eigenvalues of M. Next, we use that $\min\{\lambda, 1\} \leq \lambda$ for all real λ. Hence,

$$\operatorname{Tr}\left(\min\{\frac{t}{U}\frac{\mathbb{E}(W_n)}{\sigma^2}, 1\}\right) \leq \frac{tc}{1+\epsilon}\operatorname{Tr}\left(P_{\frac{\epsilon n\gamma}{1+\epsilon}}(K)\right).$$

Theorem B.4.1 can now be used to find how large the oversampling parameter $c > 0$ has to be so that $\lambda_{\max}(Y_n) \leq t$.

Finding a lower bound on c. In view of Theorem B.4.1, we require that $t \geq \frac{1}{c}\frac{1+\sqrt{37}}{6}$. Let the accuracy parameter be $t = 1/2$ in order to simplify the result; cfr. [165] where a similar choice is done. This implies then that $c \geq \frac{1+\sqrt{37}}{3}$. Another lower bound on c is obtained by bounding the failure probability. Namely, the failure probability given by (B.4.2) in Theorem B.4.1 is upper bounded as follows:

$$\Pr(\|X_n\|_2 > t) \leq 50\operatorname{Tr}\left(\min\{\frac{t}{U}\frac{\mathbb{E}(W_n)}{\sigma^2}, 1\}\right)e^{\frac{-t^2/2}{\sigma^2+tU/3}} \leq \frac{50tc}{1+\epsilon}d_{\text{eff}}e^{\frac{-t^2/2}{1/c+t/(3c)}},$$

where $d_{\text{eff}} = d_{\text{eff}}(\epsilon n\gamma/(1+\epsilon))$. We now find how $c > 0$ can be chosen such that this upper bound on the failure probability is smaller than δ. For $t = 1/2$ we have the condition:

$$\frac{25c}{1+\epsilon}\frac{d_{\text{eff}}}{\delta} \leq e^{\frac{c}{8(1+1/6)}}.$$

[2]Notice that σ^2 denotes here an upper bound on the predictable quadratic variation and not the bandwidth of a kernel.

Hence, by a change of variables, we can phrase this inequality in the form $\exp(x) \geq ax$, which can be written equivalently as follows:

$$x \geq -W_{-1}(-1/a), \qquad \text{for } x \geq 1 \text{ and } a \geq e,$$

where $W_{-1} : [-1/e, 0(\to \mathbb{R}$ is the negative branch of the Lambert function (see Section 3. of [48]), which is a monotone decreasing function. Notice that the asymptotic expansion of this branch of the Lambert function is $\lim_{y \to 0_-} W_{-1}(y)/\log(-y) = 1$. Therefore, we can write $-W_{-1}(-1/a) \sim \log(a)$ (with $a > 0$), so that we finally have the conditions:

$$c \geq -\frac{28}{3} W_{-1}\left(-\frac{3(1+\epsilon)\delta}{700 d_{\text{eff}}}\right), \text{ and } c \geq \frac{1+\sqrt{37}}{3},$$

where $d_{\text{eff}} = d_{\text{eff}}(\epsilon n \gamma/(1+\epsilon))$. This proves the desired result.

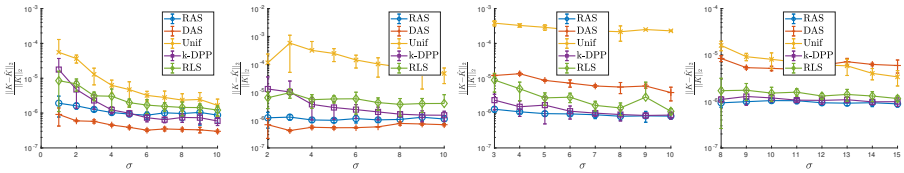
□

B.5 Ablation study for different bandwidths of the Gaussian kernel

Here we repeat the Nyström experiment of Section 3.1.6 for different bandwidths of the Gaussian kernel. Namely, we take a fixed regularization $\gamma = 10^{-4}$ in RAS, but vary the bandwidth σ of the Gaussian kernel. The parameter c is not changed and remains the same as in Table 3.1.1 for all the datasets. Note that a different σ results in a different spectrum of the kernel matrix, consequently the RAS algorithm outputs a different number of landmarks for each setting. The best performing subset is chosen as representative. The experiment is repeated 10 times and the averaged results are given in Figures B.5.1 and B.5.2. The conclusions from the experiment in Section 3.1.6 are shown to be consistent over a range of σ's.

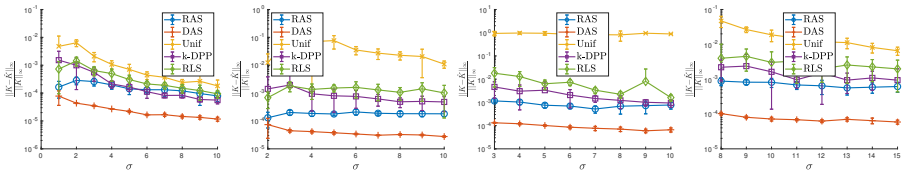
B.6 Simulation details, additional tables and figures

The computer used for the small-scale simulations has a processor with 8 cores $3.40GHz$ and 15.5 GB of RAM. The implementation of the algorithms is done with matlabR2018b. For the medium-scale experiments, the implementation of RAS was done in Julia1.0 while the simulations were performed on a computer



(a) `Stock` (b) `Housing` (c) `Abalone` (d) `Bank8FM`

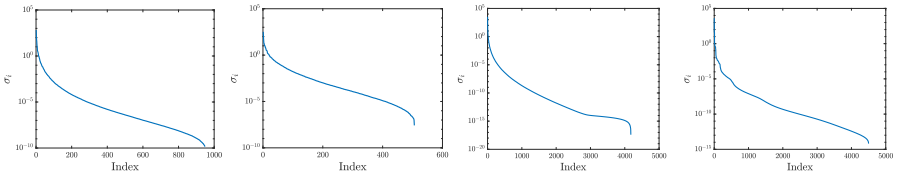
Figure B.5.1: The Relative operator norm of the Nyström approximation error as a function of the bandwidth σ. The error bars are the standard deviations.



(a) `Stock` (b) `Housing` (c) `Abalone` (d) `Bank8FM`

Figure B.5.2: The relative max norm of the Nyström approximation error as a function of the bandwidth σ. The error bars are the standard deviations.

with a 12×3.7 GHz processor and 62 GB RAM. Also, we provide here additional explanatory figures concerning the data sets of Table 3.1.1. In Figure B.6.1, the eigenvalues of the kernel matrices are displayed. The decay of the spectrum is faster for `Stock` and `Housing`. Furthermore, the accuracy of the kernel approximation for the max norm is also given in Figure B.6.3. Namely, the DAS method of Algorithm 3.1.1 shows a better approximation accuracy for this specific norm. As a measure of diversity the $\text{logdet}(K_{\mathcal{CC}}) = \sum_{i=1}^{|\mathcal{C}|} \log(\lambda_i)$ with $\lambda_1, ..., \lambda_{|\mathcal{C}|}$ the eigenvalues of $K_{\mathcal{CC}}$, is given in Figure B.6.3. The DAS method shows the largest determinant, RAS has a similar determinant as the k-DPP.

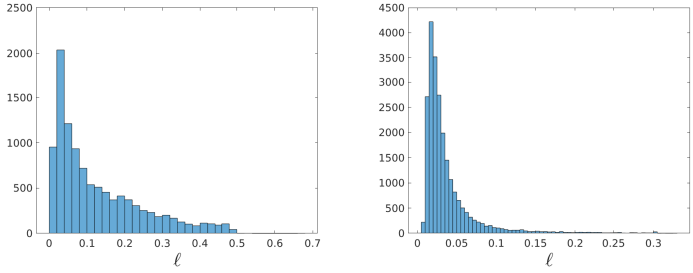


(a) `Stock` (b) `Housing` (c) `Abalone` (d) `Bank8FM`

Figure B.6.1: Singular value spectrum of the data sets of Table 3.1.1 on a logarithmic scale. For a given index (e.g., 100), the value of the eigenvalues for `Stock` and `Housing` are considerably smaller than `Abalone` and `Bank8FM`.

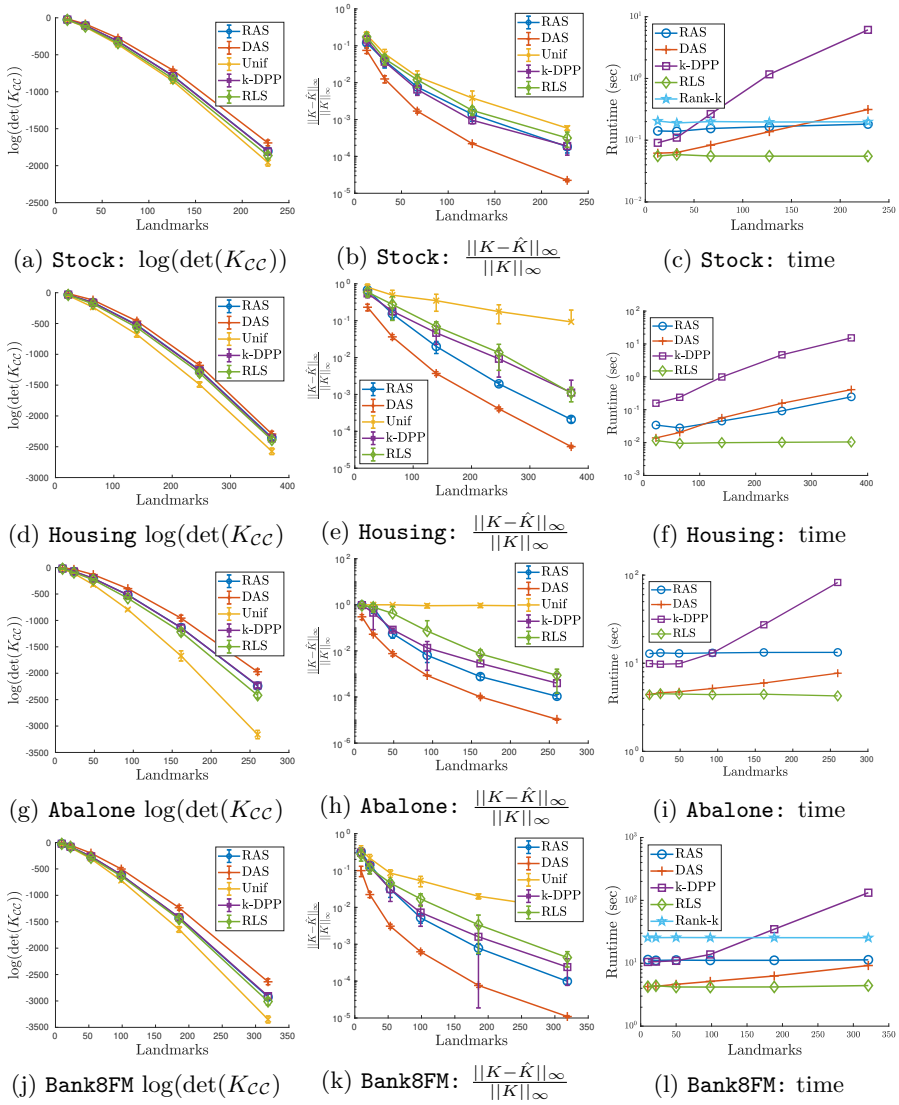
Table B.6.1: Parameters for the samplings with BLESS.

data set	σ	q_0	c_0	c_1	c_2
`Super`	3	2	2	3	3
`CASP`	1	2	2	3	3
`Adult`	10	2	2	3	2.2
`MiniBooNE`	5,10	2	2	3	3.6,4
`cod-RNA`	4	2	2	3	2.9
`Covertype`	10	2	2	3	3



(a) `Super` (b) `CASP`

Figure B.6.2: The approximate ridge leverage scores accompanying Figure 3.1.6.



(a) `Stock`: $\log(\det(K_{CC}))$ (b) `Stock`: $\frac{||K-\hat{K}||_\infty}{||K||_\infty}$ (c) `Stock`: time

(d) `Housing` $\log(\det(K_{CC})$ (e) `Housing`: $\frac{||K-\hat{K}||_\infty}{||K||_\infty}$ (f) `Housing`: time

(g) `Abalone` $\log(\det(K_{CC})$ (h) `Abalone`: $\frac{||K-\hat{K}||_\infty}{||K||_\infty}$ (i) `Abalone`: time

(j) `Bank8FM` $\log(\det(K_{CC})$ (k) `Bank8FM`: $\frac{||K-\hat{K}||_\infty}{||K||_\infty}$ (l) `Bank8FM`: time

Figure B.6.3: Diversity, relative max norm, and computation time as a function of the number of landmarks. The error bars are the standard deviations, averaged over 10 trials.

Appendix C

The Bures metric for generative adversarial networks

This appendix contains the supplementary material of the work presented in Section 4.1. In Section C.1, some details about our theoretical results are given. Next, additional experiments are described in Section C.2. Illustrative figures including generated images are listed in Section C.3. Finally, Section C.4 gives details about the architectures used in Section 4.1.

C.1 Details of the theoretical results

Let A and B be symmetric and positive semi-definite matrices. Let $A^{1/2} = U\,\mathrm{diag}(\sqrt{\boldsymbol{\lambda}})U^\top$ where U and $\boldsymbol{\lambda}$ are obtained thanks to the eigenvalue decomposition $A = U\,\mathrm{diag}(\boldsymbol{\lambda})U^\top$. We show here that the Bures distance between A and B is

$$\mathcal{B}\left(A,B\right)^2 = \min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A+B-2(A^{\frac{1}{2}}BA^{\frac{1}{2}})^{\frac{1}{2}}), \quad \text{(C.1.1)}$$

where $O(\ell)$ is the set of $\ell \times \ell$ orthonormal matrices. We can simplify the above expression as follows

$$\min_{U\in O(\ell)} \|A^{1/2} - B^{1/2}U\|_F^2 = \mathrm{Tr}(A) + \mathrm{Tr}(B) - 2\max_{U\in O(\ell)} \mathrm{Tr}(A^{1/2}B^{1/2}U) \quad \text{(C.1.2)}$$

since $\mathrm{Tr}(U^\top B^{1/2}A^{1/2}) = \mathrm{Tr}(A^{1/2}B^{1/2}U)$. Let the characteristic function of the set of orthonormal matrices be $f(U) = \chi_{O(\ell)}(U)$ that is, $f(U) = 0$ if $U \in O(\ell)$ and $+\infty$ otherwise.

Lemma C.1.2. *The Fenchel conjugate of $f(U) = \chi_{O(\ell)}(U)$ is $f^\star(M) = \|M\|_\star$, where the nuclear norm is $\|M\|_\star = \mathrm{Tr}(\sqrt{M^\top M})$ and $U, M \in \mathbb{R}^{\ell\times\ell}$.*

Proof. The definition of the Fenchel conjugate with respect to the Frobenius inner product gives

$$f^\star(M) = \sup_{U\in\mathbb{R}^{\ell\times\ell}} \mathrm{Tr}(U^\top M) - f(U) = \max_{U\in O(\ell)} \mathrm{Tr}(U^\top M).$$

Next we decompose M as follows: $M = W\Sigma V^\top$, where $W, V \in O(\ell)$ are orthogonal matrices and Σ is a $\ell \times \ell$ diagonal matrix with non negative diagonal entries, such that $MM^\top = W\Sigma^2 W^\top$ and $M^\top M = V\Sigma^2 V^\top$. Notice that the non zero diagonal entries of Σ are the singular values of M. Then,

$$\max_{U\in O(\ell)} \mathrm{Tr}(U^\top M) = \max_{U\in O(\ell)} \mathrm{Tr}(W\Sigma V^\top U) = \max_{U'\in O(\ell)} \mathrm{Tr}(\Sigma U'),$$

where we renamed $U' = V^\top UW$. Next, we remark that $\mathrm{Tr}(\Sigma U') = \mathrm{Tr}(\Sigma\,\mathrm{diag}(U'))$. Since by construction, Σ is diagonal with non negative entries the maximum is attained at $U' = \mathbb{I}$. Then, the optimal objective is $\mathrm{Tr}(\Sigma) = \mathrm{Tr}(\sqrt{M^\top M})$. □

By taking $M = A^{1/2}B^{1/2}$ we obtain (C.1.1). Notice that the role of A and B can be exchanged in (C.1.1) since U is orthogonal.

C.2 Additional experiments

Alternating training In the sequel, we also provide the results obtained thanks to Alt-BuresGAN, described in Algorithm C.2.2, which is parameter-free and yields competitive results with BuresGAN. Nevertheless, the training time of Alt-BuresGAN is larger compared to BuresGAN.

Algorithm C.2.2 Alt-BuresGAN

Sample a real and fake batch
Update G by minimizing $\mathcal{B}(\hat{C}_r, \hat{C}_g)^2$
Update G by minimizing V_G
Update D by maximizing $-V_D$;

Another matrix norm In this section, we also compare BuresGAN with a similar construction using another matrix norm. Precisely, we also use the squared Frobenius norm (Frobenius^{2}GAN), i.e., with the following generator loss $V_G + \lambda \|C_d - C_g\|_F^2$, with $\lambda = 1$. We observe that

- Frobenius^{2}GAN also performs well on simple image data sets such as Stacked MNIST,
- Frobenius^{2}GAN has a worse performance with respect to BuresGAN on CIFAR and STL-10 data sets (see Table C.2.5).

This observation emphasizes that the use of the Bures metric is requisite in order to obtain high-quality samples for higher-dimensional image data sets such as CIFAR and STL-10 and that the simpler variant based on the Frobenius norm cannot achieve a similar result.

C.2.1 Timings

The timings per iteration for the experiments presented in Section 4.1 are listed in Table C.2.1. Times are given for all the methods considered, although some methods do not always generate meaningful images for all datasets. The methods are timed for 50 iterations after the first 5 iterations, which is used as a warmup period, following which the average number of iterations per second is computed. The fastest method is the vanilla GAN. BuresGAN has a similar computation cost as GDPP. We observe that (alt-)BuresGAN is significantly faster compared to WGAN-GP. In order to obtain reliable timings, these results were obtained on the same GPU Nvidia Quadro P4000, although, for convenience, the experiments on these image datasets were executed on a machine equipped with different GPUs.

C.2.2 Best inception scores achieved with DCGAN architecture

The inception scores for the best-trained models are listed in Table C.2.2. For the CIFAR datasets, the largest inception score is significantly better than the mean for UnrolledGAN and VEEGAN. This is the same for GAN and GDPP on the STL-10 dataset, where the methods often converge to bad results. Only the proposed methods are capable of consistently generating high-quality images over all datasets.

Table C.2.1: Average time per iteration in seconds for the convolutional architecture. Average over 5 runs, with std in parenthesis. The batch size is 256. For Stacked MNIST, we use a discriminator architecture with 3 convolutional layers (see Table C.4.5). For readability, the computation times for the Grid and Ring dataset are multiplied by 10^{-3}.

	Grid	Ring	stacked MNIST	CIFAR-10	CIFAR-100	STL-10
GAN	**1.28**(0.008)	**1.28**(0.016)	**0.54**(0.0005)	**0.65**(0.02)	**0.64**(0.0008)	**6.00**(0.01)
WGAN-GP	9.0(0.04)	8.96(0.04)	2.99(0.004)	3.41(0.009)	3.41(0.006)	36.5(0.03)
UnrolledGAN	3.92(0.008)	3.92(0.028)	1.90(0.002)	2.17(0.003)	2.18(0.004)	21.99(0.06)
MDGAN-v2	3.24(0.008)	3.24(0.032)	1.66(0.002)	1.98(0.002)	1.98(0.002)	18(0.03)
VEEGAN	1.96(0.012)	1.96(0.02)	0.56(0.006)	0.66 (0.006)	0.65(0.004)	6.10(0.03)
GDPP	15.68(1.0)	16.16(0.2)	0.69(0.02)	0.80(0.02)	0.80(0.02)	7.46(0.03)
PacGAN2	1.72(0.016)	1.72(0.024)	0.77(0.006)	0.91(0.007)	0.91(0.007)	8.02(0.008)
Frobenius^{2}GAN	2.32(0.024)	2.28(0.028)	0.98(0.004)	1.09(0.004)	1.09(0.003)	13.6(0.01)
BuresGAN	11.16(0.016)	11.92(0.04)	0.72(0.02)	0.82(0.001)	0.82(0.0008)	7.6(0.03)
Alt-BuresGAN	11.92(0.024)	11.12(0.04)	0.98(0.008)	1.15(0.007)	1.15(0.007)	10.10(0.03)

Table C.2.2: Inception Score for the best trained models on CIFAR-10, CIFAR-100 and STL-10, with a DCGAN architecture (higher is better).

	CIFAR-10	CIFAR-100	STL-10
GAN	5.92	6.33	6.13
WGAN-GP	2.54	2.56	/
UnrolledGAN	4.06	4.14	/
VEEGAN	3.51	3.85	/
GDPP	6.21	6.32	6.05
BuresGAN	6.69	6.67	7.94
Alt-BuresGAN	6.40	6.48	7.88

C.2.3 Influence of the number of convolutional layers for DCGAN architecture

Results for an architecture with 3 and 4 convolutional layers described in Table C.4.5 and Table C.2.6 are listed in Section 4.1. An overview of similar experiments in other papers is given in Table C.4.3 hereafter. We provide here results with a DCGAN architecture for a simpler architecture with 2 convolutional layers for the discriminator (see, Table C.2.4 for the results and Table C.4.5 for the architecture).

Discussion. Interestingly, for most models, an improvement is observed in the quality of the images – KL divergence – and in terms of mode collapse – number of modes attained – as the size of the batch increases. For the same batch size, architecture and iterations, the image quality is improved by BuresGAN, which is robust with respect to batch size and architecture choice. The other methods show a higher variability over the different experiments. WGAN-GP has the best single run performance with a discriminator with 3 convolutional layers and has on average a superior performance when using a discriminator with 2 convolutional layers (see Table C.2.4) but sometimes fails to converge when increasing the number of discriminator layers by 1 along with increasing the batch size. MDGANv2, VEEGAN, GDPP and WGAN-GP often have an excellent single run performance. However, when increasing the number of discriminator layers, the training of these models has a tendency to collapse more often as indicated by the large standard deviation. Vanilla GAN is one of the best performing models in the variant with 3 layers. This indicates that, for certain datasets, careful architecture tuning can be more important than complicated training schemes. A lesson from Table C.2.4 is that BuresGAN's mode coverage does not vary much if the batch size increases, although the KL divergence seems to be slightly improved.

Ablation study. Results obtained with the Frobenius norm rather than the Bures metric are also included in the tables of this section.

Table C.2.3: Comparison for Frobenius norm. Experiments on the synthetic data sets. Average (std) over 10 runs. All the models are trained for 25k iterations.

	Grid (25 modes)		Ring (8 modes)	
	Nb modes	% in 3σ	Nb modes	% in 3σ
Frobenius^{2}GAN	25(0)	73(1)	8(0)	59(16)
BuresGAN	25(0)	82(1)	8(0)	82(4)
Alt-BuresGAN	**25**(0)	**84**(1)	**8**(0)	**84**(6)

Table C.2.4: KL-divergence between the generated distribution and true distribution (Quality, lower is better). The number of counted modes indicates the amount of mode collapse (higher is better). 25k iterations and average and std over 10 runs. Same architecture as in Table 2 in Section 4.1 with a discriminator with 2 convolutional layers. The architecture details are in Table C.4.5.

		Nb modes(↑)			KL div.(↓)		
	Batch size	64	128	256	64	128	256
2 conv. layers	GAN	970.5(5.8)	972.7(6.4)	979(3.5)	0.47(0.04)	0.44(0.02)	0.41(0.03)
	WGAN-GP	**996.7**(1.6)	**997.5**(0.9)	**998.1**(1.5)	**0.25**(0.01)	**0.22**(0.01)	**0.21**(0.05)
	MDGAN-v1	115.9(197)	260.9(267)	134.3(157)	5.5(1.4)	4.9(1.7)	5.8(0.9)
	MDGAN-v2	698.1(456)	898.4(299)	599.2(488)	2.2(3.0)	0.86(1.9)	2.8(3.2)
	UnrolledGAN	953.5(11)	971.4(4.8)	966.2(17.3)	0.71(0.06)	0.60(0.04)	0.58(0.10)
	VEEGAN	876.7(290)	688.5(443)	775.9(386)	0.92(1.6)	1.9(2.4)	1.54(2.2)
	GDPP	974.4(3.3)	978.2(7.6)	980.5(6.0)	0.45(0.02)	0.43(0.03)	0.41(0.03)
	PacGAN2	969.8(6.9)	971.1(3.6)	977.9(4.3)	0.54(0.04)	0.51(0.01)	0.48(0.02)
	Frobenius^{2}GAN	975.8(8.0)	981.0(4.1)	981.8(3.2)	0.34(0.03)	0.28(0.02)	0.25(0.01)
	BuresGAN	973.2(1.3)	979.9(4.0)	981.1(4.9)	0.36(0.02)	0.30(0.02)	0.25(0.01)
	Alt-BuresGAN	975.4(6.8)	978.2(5.4)	980.2(3.0)	0.37(0.02)	0.30(0.01)	0.28(0.01)

Table C.2.5: Ablation study on real image data sets for a convolutional architecture. Frobenius norm is also considered in the place of the Bures metric. Generation quality on CIFAR-10, CIFAR-100, and STL-10 with a DCGAN architecture. Average(std) over 10 runs. 100k iterations for each. For improving readability, the SWD score was multiplied by 100.

	CIFAR-10			CIFAR-100			STL-10		
	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)	IS(↑)	FID(↓)	SWD(↓)
Frobenius^{2}GAN	5.40(0.16)	57.7(4.8)	**1.3**(0.3)	5.52(0.16)	60.1(4.4)	**1.6**(0.5)	6.26(0.15)	132.18(3.6)	**1.8**(0.4)
BuresGAN	**6.34**(0.17)	**43.7**(0.9)	2.1(0.6)	**6.5**(0.1)	**47.2**(1.2)	2.1(1.0)	**7.6**(0.3)	**109**(7)	2.3(0.3)
Alt-BuresGAN	6.23(0.07)	45.4(2.8)	1.7(0.9)	6.4(0.1)	49.4(3.4)	1.8(0.6)	7.5(0.3)	110(4)	2.8(0.4)

C.3 Additional figures

Next, in Figure C.3.1, Figure C.3.2, Figure C.3.3, and Figure C.3.4, we list samples of generated images for the models trained with a DCGAN architecture for Stacked MNIST, CIFAR-10, CIFAR-100, and STL-10 respectively.

Table C.2.6: Ablation study for Stacked MNIST experiment for an architecture with 4 convolutional layers (see Table C.4.4). All the models are trained for 25k iterations with a batch size of 64, a learning rate of 2e−4 for Adam, and a normal latent distribution. The evaluation is over 10k samples and we report an average (std) over 10 runs.

	Nb modes(↑)	KL div.(↓)
Frobenius^{2}GAN	984.7(6.4)	0.38(0.04)
BuresGAN	989.9(4.7)	0.38(0.06)
Alt-BuresGAN	990.0(4.9)	0.33(0.04)

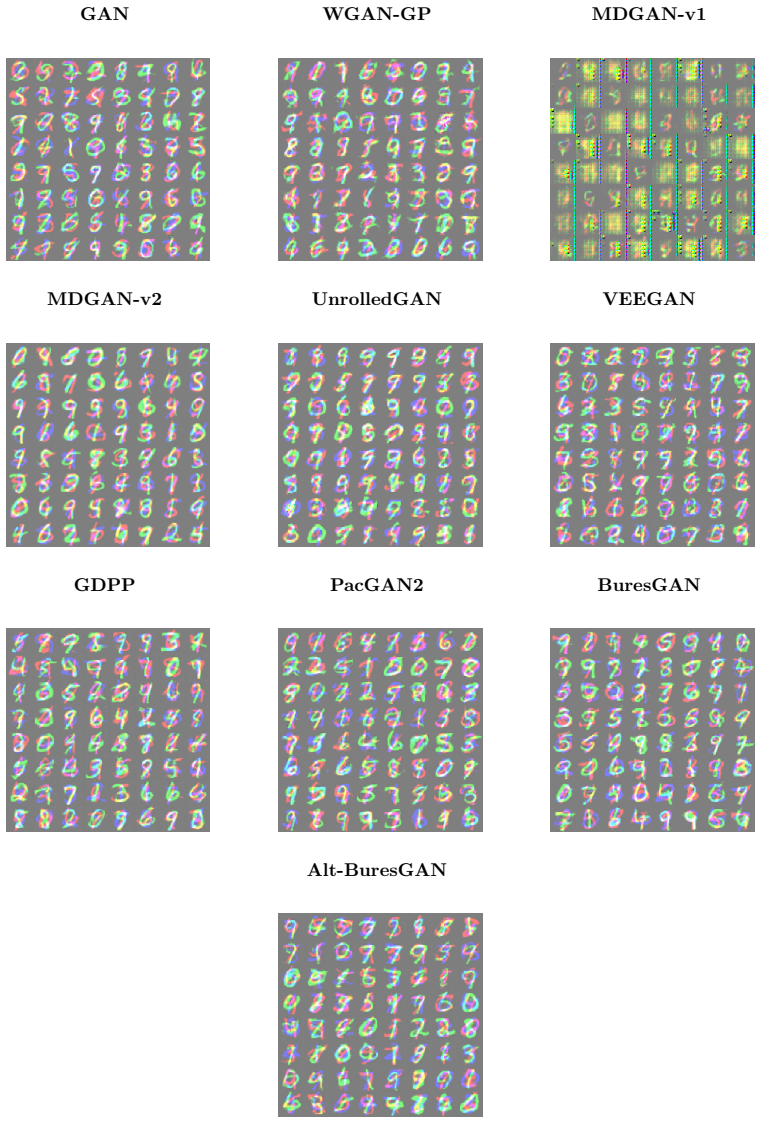


Figure C.3.1: Generated images for the Stacked MNIST dataset. Each model is trained with 3 layers and mini-batch size 256. Each square shows 64 samples from the trained generator.



Figure C.3.2: Generated images for CIFAR-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator.



Figure C.3.3: Generated images for CIFAR-100 using a DCGAN architecture. Each square shows 64 samples from the trained generator.

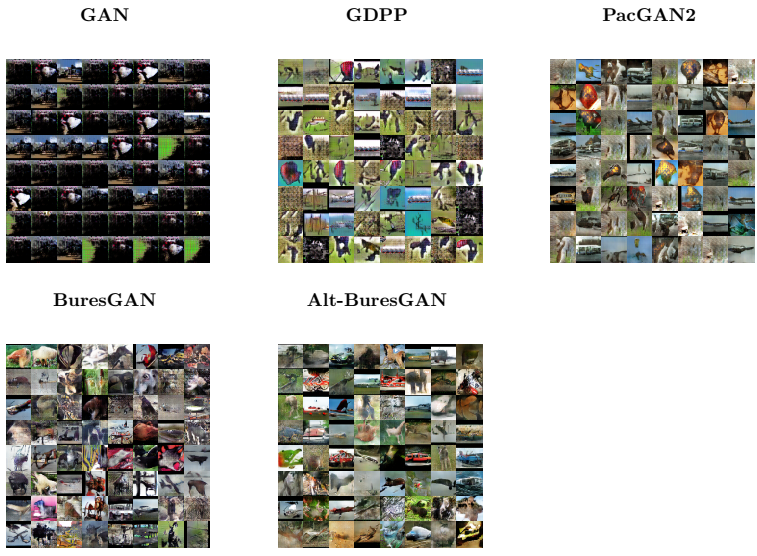


Figure C.3.4: Generated images for STL-10 using a DCGAN architecture. Each square shows 64 samples from the trained generator.

C.4 Architectures

C.4.1 Synthetic architectures

Following the recommendation in the original work of [217], the same fully connected architecture is used for the VEEGAN reconstructor in all experiments.

Table C.4.1: The generator and discriminator architectures for the synthetic examples.

Layer	Output	Activation
Input	256	-
Dense	128	tanh
Dense	128	tanh
Dense	2	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	1	-

Table C.4.2: Respectively the MDGAN encoder model and VEEGAN stochastic inverse generator architectures for the synthetic examples. The output of the VEEGAN models are samples drawn from a normal distribution with a scale of 1 and where the location is learned.

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	256	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	256	tanh
Normal	256	-

Stacked MNIST Architectures For clarity, we report in Table C.4.3 the settings of different papers for the Stacked MNIST experiment. This helps to clarify the difference between our simulations and other papers' simulations. On the one hand, in Table C.4.4, the architecture of the experiment reported in Section 4.1 is described. On the other hand, the architecture of the supplementary experiments on Stacked MNIST is given in Table C.4.5. Details of the architecture of MDGAN are also reported in Table C.4.6.

Table C.4.3: The hyperparameters and architectures of the stacked MNIST experiment performed by the different methods are given here. This represents our best efforts at providing a comparison between the different parameters used. The asterisks ** indicate that the parameters were obtained by the Github repository. Notice that GDPP paper also used 30000 iterations for training DCGAN and unrolled GAN (indicated by *). BuresGAN's column refers to the settings of the experiment in Table 2 in Section 4.1 and Table C.2.6 respectively, for which the different values are separated by the symbol &.

Parameters	BuresGAN	PacGAN	VEEGAN	GDPP
learning rates	1×10^{-3} & 2×10^{-4}	2×10^{-4}	2×10^{-4}	1×10^{-4} **
learning rate decay	no	no	no	no**
Adam β_1	0.5	0.5	0.5	0.5
Adam β_2	0.999	0.999	0.999	0.9
iterations	25000	20000	?	15000*
disc. conv. layers	3 & 4	4	4	3
gen. conv. layers	3 & 4	4	4	3
batch size	64, 128, 256 & 64	64	64	64
evaluation samples	10000	26000	26000	26000
ℓ (i.e., z dimension)	100	100	100	128
z distribution	$\mathcal{N}(0, \mathbb{I}_\ell)$	unif$[-1,1]^\ell$	unif$[-1,1]^\ell$ **	unif$[-1,1]^\ell$

Table C.4.4: The generator and discriminator architectures for the Stacked MNIST experiments with 4 layers for both generator and discriminator. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	2048	ReLU	Yes
Reshape	2, 2, 512	-	-
Conv'	4, 4, 256	ReLU	Yes
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Conv	2, 2, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table C.4.5: The generator and discriminator architectures for the Stacked MNIST experiments. The BN column indicates whether batch normalization is used after the layer or not. For the experiments with 2 convolution layers in Table C.2.4, the final convolution layer is removed in the discriminator.

Layer	Output	Activation	BN
Input	100	-	-
Dense	12544	ReLU	Yes
Reshape	7, 7, 256	-	-
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table C.4.6: Model architecture for the Stacked MNIST experiments. (Left) The CNN architecture of the classifier used during the evaluation. Dropout with a rate of 0.5 is used before the final dense layer. (Right) The MDGAN encoder model architecture. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation
Input	28, 28,1	-
Conv	24, 24, 32	ReLU
MaxPool	12, 12, 32	-
Conv	8, 8, 64	ReLU
MaxPool	4, 4, 64	-
Flatten	-	-
Dense	1024	ReLU
Dense	10	-

Layer	Output	Activation	BN
Input	28, 28, 3	-	-
Conv	14, 14, 3	ReLU	Yes
Conv	7, 7, 64	ReLU	Yes
Conv	7, 7, 128	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

C.4.2 CIFAR-10 and 100 DCGAN architectures

Convolutional architectures for CIFAR data set are reported in Table C.4.7 and Table C.4.8 below.

Table C.4.7: The generator and discriminator architectures for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	16384	ReLU	Yes
Reshape	8, 8, 256	-	-
Conv'	8, 8, 128	ReLU	Yes
Conv'	16, 16, 64	ReLU	Yes
Conv'	32, 32, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	32, 32, 3	-	-
Conv	16, 16, 64	Leaky ReLU	No
Conv	8, 8, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table C.4.8: The MDGAN encoder model architecture for the CIFAR-10 and CIFAR-100 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	32, 32, 3	-	-
Conv	16, 16, 3	ReLU	Yes
Conv	8, 8, 64	ReLU	Yes
Conv	8, 8, 128	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

C.4.3 STL-10 DCGAN architectures

In Table C.4.9, the convolutional architecture is described for the $96 \times 96 \times 3$ STL-10 data set, while MDGAN details are reported in Table C.4.10.

Table C.4.9: The generator and discriminator architectures for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	100	-	-
Dense	36864	ReLU	Yes
Reshape	12, 12, 256	-	-
Conv'	12, 12, 256	ReLU	Yes
Conv'	24, 24, 128	ReLU	Yes
Conv'	48, 48, 64	ReLU	Yes
Conv'	96, 96, 3	ReLU	Yes

Layer	Output	Activation	BN
Input	96, 96, 3	-	-
Conv	48, 48, 64	Leaky ReLU	No
Conv	24, 24, 128	Leaky ReLU	Yes
Conv	12, 12, 256	Leaky ReLU	Yes
Conv	6, 6, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table C.4.10: The MDGAN encoder model architecture for the STL-10 experiments. The BN column indicates whether batch normalization is used after the layer or not.

Layer	Output	Activation	BN
Input	96, 96, 3	-	-
Conv	48, 48, 3	ReLU	Yes
Conv	24, 24, 64	ReLU	Yes
Conv	12, 12, 128	ReLU	Yes
Conv	12, 12, 256	ReLU	Yes
Flatten	-	-	-
Dense	100	-	-

C.4.4 ResNet architectures

For CIFAR-10, we used the ResNet architecture from the appendix of [91] with minor changes as given in Table C.4.11. We used an initial learning rate of 5e−4 for CIFAR-10 and STL-10. For both datasets, the models are run for 200k iterations. For STL-10, we used a similar architecture that is given in Table C.4.11.

Table C.4.11: The generator (first) and discriminator (second) ResNet architectures for the CIFAR-10 experiments. The generator (third) and discriminator (fourth) ResNet architectures for the STL-10 experiments. For the experiment with full-sized 96x96 images, an extra upsampling block was added to the generator.

Layer	Kernel Size	Resample	Output Shape
Input	-	-	128
Dense	-	-	$200 \cdot 4 \cdot 4$
Reshape	-	-	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 8 \times 8$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 16 \times 16$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 32 \times 32$
Conv, tanh	3×3	-	$3 \times 32 \times 32$

Layer	Kernel Size	Resample	Output Shape
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 16 \times 16$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 8 \times 8$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 4 \times 4$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 4 \times 4$
ReLu, meanpool	-	-	200
Dense	-	-	1

Layer	Kernel Size	Resample	Output Shape
Input	-	-	128
Dense	-	-	$200 \cdot 6 \cdot 6$
Reshape	-	-	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	up	$200 \times 48 \times 48$
Conv, tanh	3×3	-	$3 \times 48 \times 48$

Layer	Kernel Size	Resample	Output Shape
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 24 \times 24$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 12 \times 12$
ResBlock	$[3 \times 3] \times 2$	Down	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 6 \times 6$
ResBlock	$[3 \times 3] \times 2$	-	$200 \times 6 \times 6$
ReLu, meanpool	-	-	200
Dense	-	-	1

Appendix D

Leverage score sampling for complete mode coverage in generative adversarial networks

This appendix contains the supplementary material of the work presented in Section 4.2. In Section D.1, the remaining results from the UNBALANCED MNIST experiment are given. In Section D.2, the effect of the ridge parameter γ and dimensionality reduction size k is discussed. Then, in Section D.3, the training times of our simulations are reported. Section D.4 provides further information about the distribution of generated samples in the artificial datasets. Finally, Section D.5 describes the experimental setting, the proposed algorithm, and architectures used in the Section 4.2.

D.1 Extra table unbalanced MNIST dataset

For the UNBALANCED MNIST experiment, only the number of samples in the minority modes are given in the main part. The number of samples for the remaining modes is given in Table D.1.1.

Table D.1.1: Experiments on the UNBALANCED MNIST dataset. Two variants of RLS BuresGAN are trained. First, RLSs are calculated with an explicit feature map obtained from a pre-trained classifier (Class.). The second version uses the next-to-last layer of the discriminator (Discr.) to compute the RLSs. The RLS MwuGAN is initialized using the RLSs with the explicit feature maps obtained from a pre-trained classifier.

	Mode 6	Mode 7	Mode 8	Mode 9	Mode 10
GAN	1831(135)	1835(32)	2009(69)	1747(46)	1915(143)
PacGAN2	1767(116)	1864(16)	1983(105)	1847(25)	1900(110)
BuresGAN	1762(96)	1870(38)	1979(110)	1782(29)	1916(68)
IwGAN	1833(74)	1837(97)	1902(79)	1783(64)	1928(74)
IwMmdGAN	67(45)	0(0)	4(3)	8741(430)	22(10)
MwuGAN (15)	1638(137)	1986(114)	1807(96)	2026(22)	1801(123)
RLS MwuGAN (15) (ours)	1616(17)	1513(144)	2018(78)	1576(30)	1811(34)
RLS BuresGAN Class. (ours)	1311(178)	1137(116)	1743(79)	1069(67)	1396(100)
RLS BuresGAN Discr. (ours)	2012(97)	1839(53)	1608(155)	1841(55)	1545(115)

D.2 Effect of the regularization parameter in RLS

D.2.1 Synthetic data

We display in Table D.2.1 an ablation study for different ridge regularization parameters $\gamma > 0$ on the synthetic RING and GRID. A lesson from Table D.2.1 is that the performance on the synthetic datasets does not vary much if different values of γ are chosen in the case of the discriminator feature map. Notice that, for $\gamma = 10^{-3}$ and $\gamma = 10^{-4}$, there is a small improvement in sample quality for RING and in terms of mode coverage for GRID with the Gaussian kernel feature map.

Impact of the number of generators in MwuGAN. RLSs can also be used as initial starting weights in MwuGAN. The RLSs are constructed with a Gaussian kernel with bandwidth $\sigma = 0.15$ and regularization $\gamma = 10^{-3}$. Both methods contain a mixture of 15 GANs. This approach is compared with the classical initialization with uniform weights. In Figure D.2.1, a clear improvement over a uniform initialization is visible when RLSs are used as initial weights.

Table D.2.1: Ablation study over the parameter γ on the synthetic datasets for RLS BuresGAN with a discriminator feature map (Discr.) and Gaussian feature map (Gauss.) with bandwidth $\sigma = 0.15$. No significant difference can be seen from the reported performance as the regularization parameter varies.

Feature map	γ	Ring with 8 modes		Grid with 25 modes	
		Nb modes	% in 3σ	Nb modes	% in 3σ
Discr.	10^{-2}	8(0)	0.92(0.02)	24.0(1.3)	0.79(0.03)
	10^{-3}	8(0)	0.90(0.02)	24.4(0.9)	0.78(0.06)
	10^{-4}	8(0)	0.90(0.01)	24.5(0.8)	0.77(0.07)
Gauss.	10^{-2}	8(0)	0.83(0.10)	19.9(1.9)	0.76(0.09)
	10^{-3}	8(0)	0.90(0.02)	24.0(1.5)	0.76(0.11)
	10^{-4}	8(0)	0.87(0.04)	24.8(0.6)	0.78(0.03)

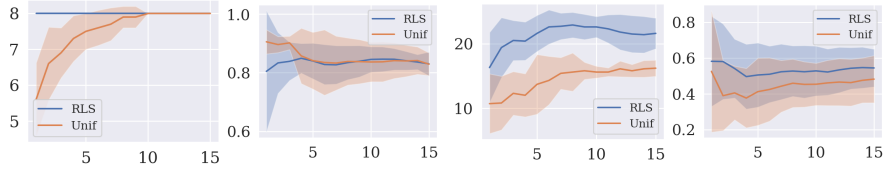


Figure D.2.1: RING: number of modes (first) and sample quality (second). GRID: number of modes (third) and sample quality (fourth). The x-axis refers to the number of generators of MwuGAN. Sample quality is assessed by counting the number of modes within 3σ of each mode. The blue (resp. red) curve indicates the results obtained for RLS (resp. unif.) sampling with MwuGAN.

D.2.2 Unbalanced MNIST

We describe here an ablation study for different ridge regularization parameters $\gamma > 0$ and size of dimension reduction k on the UNBALANCED 012-MNIST and UNBALANCED MNIST dataset. Moreover, the use of sketching the fixed explicit feature map is analyzed. From these results is concluded that the non-linear UMAP procedure leads to better mode coverage. The results are given in Tables D.2.2 and D.2.3.

Table D.2.2: Ablation study over the regularization parameter γ and dimension k on the UNBALANCED 012-MNIST dataset for RLS BuresGAN with a discriminator feature map (Discr.) with sketching and an explicit feature map obtained from the next-to-last layer of a pre-trained classifier (Class.) with a UMAP dimensionality reduction and sketched dimensionality reduction. Minority modes are highlighted in black in the first row. The best performance is achieved using the Discr. sketched and Class. UMAP. feature maps with regularization 10^{-4} and dimension $k = 25$.

Feature map	Dim.	γ	Mode 1	Mode 2	**Mode 3**	KL
Discr. Sketched	10	10^{-2}	5561(381)	2756(300)	1409(345)	0.16(0.05)
		10^{-3}	5568(298)	2769(178)	1386(198)	0.15(0.03)
		10^{-4}	5655(279)	2812(135)	1218(253)	0.17(0.04)
	25	10^{-2}	5814(379)	2405(225)	1501(247)	0.17(0.04)
		10^{-3}	5715(312)	2477(139)	1415(310)	0.17(0.04)
		10^{-4}	5748(172)	2416(268)	1566(293)	0.16(0.02)
Class. UMAP	10	10^{-2}	3711(173)	5361(103)	751(161)	0.18(0.03)
		10^{-3}	3813(195)	5425(208)	644(67)	0.19(0.01)
		10^{-4}	3713(221)	5431(226)	712(152)	0.18(0.03)
	25	10^{-2}	4221(104)	4958(137)	672(109)	0.18(0.02)
		10^{-3}	4179(111)	4876(148)	802(105)	0.16(0.01)
		10^{-4}	3414(161)	4862(134)	1461(183)	0.08(0.01)
Class. Sketched	10	10^{-2}	5865(122)	3389(79)	605(111)	0.25(0.03)
		10^{-3}	5715(241)	3456(216)	683(161)	0.31(0.01)
		10^{-4}	5952(309)	3403(206)	513(135)	0.27(0.04)
	25	10^{-2}	5998(193)	3130(105)	672(119)	0.25(0.03)
		10^{-3}	5957(100)	3127(136)	775(95)	0.23(0.01)
		10^{-4}	6009(157)	3138(142)	698(121)	0.32(0.01)

Table D.2.3: Ablation study over the regularization parameter γ and dimension k on the UNBALANCED MNIST dataset for RLS BuresGAN with a discriminator feature map (Discr.) with sketching and an explicit feature map obtained from the next-to-last layer of a pre-trained classifier (Class.) with a UMAP dimensionality reduction and sketched dimensionality reduction. Minority modes are highlighted in black in the first row. The best performance is achieved using the Class. UMAP. feature map.

Feature map	Dim.	γ	**Mode 1**	**Mode 2**	**Mode 3**	**Mode 4**	**Mode 5**	Mode 6	Mode 7	Mode 8	Mode 9	Mode 10	KL
Discr. Sketched	10	10^{-2}	254(77)	170(30)	222(40)	227(46)	212(62)	2061(94)	1871(134)	1598(71)	1853(83)	1532(96)	0.38(0.02)
		10^{-3}	214(38)	172(123)	239(35)	223(99)	230(54)	2081(91)	1832(32)	1611(100)	1898(44)	1501(123)	0.39(0.02)
		10^{-4}	235(62)	183(141)	264(44)	255(109)	219(54)	2012(97)	1839(53)	1608(155)	1841(55)	1545(115)	0.37(0.02)
	25	10^{-2}	236(56)	191(145)	286(43)	231(72)	219(60)	2066(113)	1765(48)	1505(81)	1958(37)	1543(56)	0.37(0.02)
		10^{-3}	230(30)	172(129)	267(31)	248(89)	274(68)	2046(100)	1780(42)	1546(122)	1899(73)	1538(135)	0.37(0.02)
		10^{-4}	224(49)	147(179)	255(27)	223(52)	197(53)	2032(116)	1790(49)	1554(119)	1962(55)	1617(143)	0.40(0.02)
Class. UMAP	10	10^{-2}	885(79)	662(143)	359(229)	814(97)	578(66)	1248(152)	1137(152)	1847(154)	1095(98)	1374(112)	0.10(0.01)
		10^{-3}	892(137)	674(90)	370(227)	803(108)	605(75)	1187(189)	1123(103)	1784(111)	1110(95)	1452(189)	0.09(0.02)
		10^{-4}	875(112)	663(122)	360(198)	831(59)	615(82)	1311(178)	1137(116)	1743(79)	1069(67)	1396(100)	0.09(0.01)
	25	10^{-2}	556(60)	525(82)	352(59)	542(71)	473(56)	1471(99)	1449(87)	1915(101)	1327(66)	1392(80)	0.16(0.01)
		10^{-3}	495(58)	473(114)	385(50)	546(83)	463(50)	1450(120)	1452(69)	1964(110)	1290(75)	1483(81)	0.17(0.02)
		10^{-4}	499(73)	443(103)	358(48)	513(104)	450(45)	1491(148)	1468(51)	1957(119)	1329(43)	1493(145)	0.18(0.01)
Class. Sketched	10	10^{-2}	218(72)	188(171)	165(31)	261(114)	246(41)	2019(166)	1726(48)	1859(204)	1683(63)	1637(178)	0.38(0.02)
		10^{-3}	189(45)	194(238)	184(88)	245(232)	221(44)	2147(176)	1712(38)	1824(182)	1666(61)	1618(177)	0.4(0.03)
		10^{-4}	239(45)	127(200)	211(27)	232(133)	214(65)	2127(113)	1783(48)	1804(122)	1642(29)	1622(142)	0.48(0.02)
	25	10^{-2}	238(30)	192(205)	189(35)	308(84)	254(39)	2011(105)	1699(74)	1710(127)	1785(54)	1615(143)	0.36(0.02)
		10^{-3}	262(45)	204(116)	251(45)	258(83)	266(62)	2041(152)	1621(40)	1774(119)	1751(41)	1573(134)	0.35(0.02)
		10^{-4}	246(50)	220(131)	199(41)	312(70)	306(41)	2033(127)	1689(38)	1712(115)	1724(67)	1560(100)	0.34(0.02)

D.3 Timings

Training times with a single NVIDIA Tesla P100-SXM2-16GB @1.3 GHz GPU. Temporary evaluations and plotting during training are included in the timings, so only relative comparisons are instructive.

Table D.3.1: Total training time in seconds, averaged over 10 runs.

	Ring	Grid	012-MNIST	MNIST
GAN	45(1)	48(1)	1063(4)	1074(3)
PacGAN2	65(1)	71(1)	1205(4)	1211(8)
BuresGAN	100(1)	105(2)	1357(3)	1373(3)
IwGAN	65(1)	65(1)	1066(5)	1080(3.3)
IwMmdGAN	45(1)	56(1)	519(4)	5126(1)
MwuGAN (15)	1195(9)	1225(8)	20730(314)	36074(292)
RLS MwuGAN (15)	1198(9)	1217(9)	20823(233)	35400(566)
RLS BuresGAN Gauss.	130(3)	136(2)	/	/
RLS BuresGAN Class.	/	/	1615(11)	2494(8)
RLS BuresGAN Discr.	176(4)	187(1)	2006(3)	2004(2)

D.4 Distribution of generated samples for the artificial datasets

The number of samples per mode for all GANs in the Ring experiments are given in Figure D.4.1. The number of samples per mode for all GANs in the Grid experiments are given in Figure D.4.2.

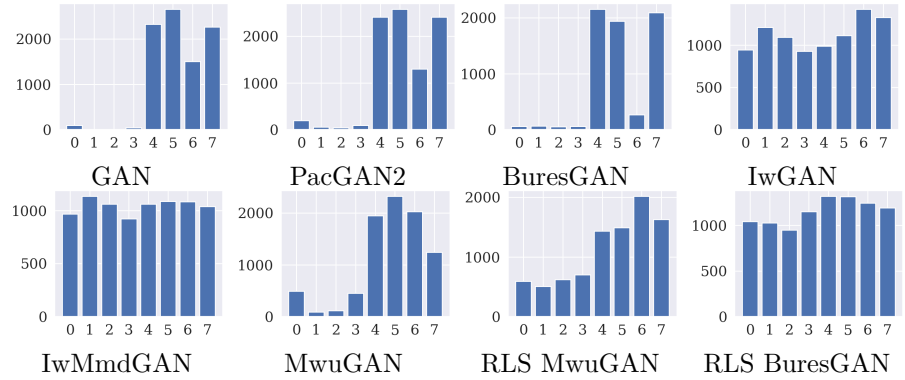


Figure D.4.1: Samples per mode for a single GAN trained on the Ring dataset. Only IwMmdGAN, MwuGAN (15), RLS MwuGAN (15), and RLS BuresGAN with a discriminator feature map (Discr.) are capable of covering all the modes.

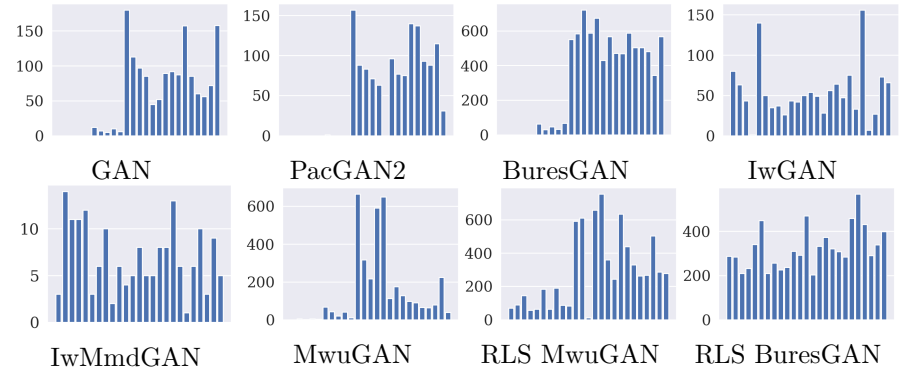


Figure D.4.2: Samples per mode for a single GAN trained on the Grid dataset. Only the proposed RLS BuresGAN with a discriminator feature map (Discr.) is capable of covering all the modes.

D.5 Architectures, algorithms, and settings

Settings experimental section. Unless specified otherwise, the models are trained for 30k iterations with a batch size of 64, by using the Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate 10^{-3} for both the generator and discriminator. The dimensionality of the generator latent space ℓ is equal to 100. The results of the experimental section always refer to the performance achieved at the end of the training. All the images are scaled in between -1

and 1 before running the algorithms. The hyperparameters of the competing methods are chosen as suggested in the authors' reference implementations. For MwuGAN, we take $\delta = 0.25$ and run for an increasing number of generators (displayed in brackets). In all the simulations, PacGAN always uses 2 as packing number. Note that IwMmdGAN and IwGAN require full knowledge of the Radon-Nikodym derivative M, where we want to generate data from a target distribution p but only have access to representative samples from a modified distribution Mp. The method is thus not an unsupervised method and tackles a slightly different problem. However, the method was included for completeness.

Algorithm D.5.1 RLS BuresGAN with a fixed feature map.

Input: $\{\boldsymbol{x}_i\}_{i=1}^n$, regularization γ, dimension reduction size k, discriminator D, generator G, feature map $\varphi(\cdot) \in \mathbb{R}^m$ - explicit *or* implicit via kernels $k(\cdot, \cdot)$

Calculate ridge leverage scores.

if $\varphi(\cdot)$ = Implicit **then**
    Calculate the RLSs for all points: $\ell_i = \left(K(K + n\gamma\mathbb{I})^{-1}\right)_{ii}$.
else if $\varphi(\cdot)$ = Explicit **then**
    if dimension reduction **then**
        $\varphi(\boldsymbol{x}_i) \leftarrow$ UMAP of size k calculated on the full projected dataset $\{\varphi(\boldsymbol{x}_i)\}_{i=1}^n$.
    else
        $k \leftarrow m$.
    end if
    if $n > k$ **then**
        Calculate the RLSs for all points: $\ell_i = \varphi(\boldsymbol{x}_i)^\top (C + n\gamma\mathbb{I})^{-1}\varphi(\boldsymbol{x}_i)$.
    else
        Calculate the RLSs for all points: $\ell_i = \left(K(K + n\gamma\mathbb{I})^{-1}\right)_{ii}$.
    end if
end if

Train BuresGAN.

while not converged **do**
    Sample a real and fake batch with probability $\boldsymbol{x}_i \sim \ell_i$.
    Update the generator G by minimizing $V_G + \mathcal{B}(\hat{C}_r, \hat{C}_g)^2$
    Update the discriminator D by maximizing $-V_D$
end while

Algorithm D.5.2 RLS BuresGAN with a discriminator-based explicit feature map.

Input: $\{\boldsymbol{x}_i\}_{i=1}^n$, regularization γ, dimension reduction size k, mini-batch size s, discriminator D with discriminator feature map $\varphi_D(\cdot) \in \mathbb{R}^m$, generator G.
Train BuresGAN with adaptive RLSs.

while not converged **do**
  Sample a mini-batch $\hat{\boldsymbol{x}}$ uniform at random of size $b = 20s$.
  if dimension reduction **then**
    $\varphi(\hat{\boldsymbol{x}}_i) \leftarrow S^\top \varphi_D(\hat{\boldsymbol{x}}_i) \in \mathbb{R}^k$, with sketching matrix $S = A/\sqrt{k} \in \mathbb{R}^{m\times k}$.
  else
    $\varphi(\hat{\boldsymbol{x}}_i) \leftarrow \varphi_D(\hat{\boldsymbol{x}}_i)$, $k \leftarrow m$.
  end if
  if $b > k$ **then**
    Calculate the RLSs for all points: $\ell_i = \varphi(\boldsymbol{x}_i)^\top (C + n\gamma\mathbb{I})^{-1}\varphi(\boldsymbol{x}_i)$.
  else
    Calculate the RLSs for all points: $\ell_i = \left(K(K + n\gamma\mathbb{I})^{-1}\right)_{ii}$.
  end if
  Sample a real and fake batch of size s with probability $\hat{\boldsymbol{x}}_i \sim \ell_i$.
  Update the generator G by minimizing $V_G + \mathcal{B}(\hat{C}_r, \hat{C}_g)^2$
  Update the discriminator D by maximizing $-V_D$
end while

Table D.5.1: The generator (left) and discriminator (right) architectures for the synthetic examples.

Layer	Output	Activation
Input	25	-
Dense	128	tanh
Dense	128	tanh
Dense	2	-

Layer	Output	Activation
Input	2	-
Dense	128	tanh
Dense	128	tanh
Dense	1	-

Table D.5.2: The generator (left) and discriminator (right) architectures for the UNBALANCED 012-MNIST and the UNBALANCED MNIST experiments. BN indicates if batch normalization is used.

Layer	Output	Activation	BN
Input	100	-	-
Dense	12544	ReLU	Yes
Reshape	7, 7, 256	-	-
Conv'	7, 7, 128	ReLU	Yes
Conv'	14, 14, 64	ReLU	Yes
Conv'	28, 28, 1	ReLU	Yes

Layer	Output	Activation	BN
Input	28, 28, 1	-	-
Conv	14, 14, 64	Leaky ReLU	No
Conv	7, 7, 128	Leaky ReLU	Yes
Conv	4, 4, 256	Leaky ReLU	Yes
Conv	2, 2, 512	Leaky ReLU	Yes
Flatten	-	-	-
Dense	1	-	-

Table D.5.3: The CNN architecture of the classifier used during the evaluation of the MNIST experiments. Dropout with a rate of 0.5 is used before the final dense layer.

Layer	Output	Activation
Input	28, 28,1	-
Conv	24, 24, 32	ReLU
MaxPool	12, 12, 32	-
Conv	8, 8, 64	ReLU
MaxPool	4, 4, 64	-
Flatten	-	-
Dense	1024	ReLU
Dense	10	-

Appendix E

Generative restricted kernel machines: a framework for multi-view generation and disentangled feature learning

This appendix contains the supplementary material of the work presented in Section 5.2. In Section E.1, we give the derivation for computing latent variables using covariance matrix. In Section E.2, the proofs regarding the stabilization term are given. Section E.3 includes the parameters and architectures used in the experiments. Finally, Section E.4 contains extra experiments and visualizations.

E.1 Computing latent variables using covariance matrix

From (5.2.2), eliminating the variables $\boldsymbol{h}_i$ yields the following:

$$\frac{1}{\eta_1}\left[\sum_{i=1}^{N}\phi_1(\boldsymbol{x}_i)\phi_1(\boldsymbol{x}_i)^\top U + \sum_{i=1}^{N}\phi_1(\boldsymbol{x}_i)\phi_2(\boldsymbol{y}_i)^\top V\right] = \Lambda U,$$

$$\frac{1}{\eta_2}\left[\sum_{i=1}^{N}\phi_2(\boldsymbol{y}_i)\phi_1(\boldsymbol{x}_i)^\top U + \sum_{i=1}^{N}\phi_2(\boldsymbol{y}_i)\phi_2(\boldsymbol{y}_i)^\top V\right] = \Lambda V.$$

Denote $\phi_{\boldsymbol{x}} \coloneqq [\phi_1(\boldsymbol{x}_1), \ldots, \phi_1(\boldsymbol{x}_N)]$, $\phi_{\boldsymbol{y}} \coloneqq [\phi_2(\boldsymbol{y}_1), \ldots, \phi_2(\boldsymbol{y}_N)]$ and the diagonal matrix $\Lambda = \text{diag}\{\lambda_1, \ldots, \lambda_s\} \in \mathbb{R}^{s\times s}$ with $s \leq N$. Now, composing the above equations in matrix form, we get the following eigen-decomposition problem:

$$\begin{bmatrix} \frac{1}{\eta_1}\phi_{\boldsymbol{x}}\phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_1}\phi_{\boldsymbol{x}}\phi_{\boldsymbol{y}}^\top \\ \frac{1}{\eta_2}\phi_{\boldsymbol{y}}\phi_{\boldsymbol{x}}^\top & \frac{1}{\eta_2}\phi_{\boldsymbol{y}}\phi_{\boldsymbol{y}}^\top \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} \Lambda.$$

Here the size of the covariance matrix is $(d_f+p_f)\times(d_f+p_f)$. The latent variables $\boldsymbol{h}_i$ can be computed using (5.2.2), which simply involves matrix multiplications.

E.2 Stabilizing the objective function

Proposition E.2.1. *All stationary solutions for $\boldsymbol{h}$, Λ in (5.2.3) of $\mathcal{J}_t$ lead to $\mathcal{J}_t = 0$.*

Proof. Let $\lambda_i, \boldsymbol{h}_i$ are given by (5.2.3). Using (5.2.2) to substitute V and U in (5.2.1) yields:

$$\begin{aligned}
\mathcal{J}_t(V,U,\Lambda,h) &= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{\eta_1}{2}\,\text{Tr}\left(\frac{1}{\eta_1^2}\sum_{i=1}^{N}\boldsymbol{h}_i\phi_1(\boldsymbol{x}_i)^\top \sum_{j=1}^{N}\phi_1(\boldsymbol{x}_j)\boldsymbol{h}_j^\top\right) \\
&\quad + \frac{\eta_2}{2}\,\text{Tr}\left(\frac{1}{\eta_2^2}\sum_{i=1}^{N}\boldsymbol{h}_i\phi_2(\boldsymbol{y}_i)^\top \sum_{j=1}^{N}\phi_2(\boldsymbol{y}_j)\boldsymbol{h}_j^\top\right) \\
&= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{\eta_1}{2}\,\text{Tr}\left(\frac{1}{\eta_1^2}\boldsymbol{h}K_1\boldsymbol{h}^\top\right) + \frac{\eta_2}{2}\,\text{Tr}\left(\frac{1}{\eta_2^2}\boldsymbol{h}K_2\boldsymbol{h}^\top\right) \\
&= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{1}{2}\,\text{Tr}\left(\boldsymbol{h}\left[\frac{1}{\eta_1}K_1 + \frac{1}{\eta_2}K_2\right]\boldsymbol{h}^\top\right).
\end{aligned}$$

From (5.2.3), we get:

$$\begin{aligned}
\mathcal{J}_t(V,U,\Lambda,h) &= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{1}{2}\,\text{Tr}\left(\boldsymbol{h}\boldsymbol{h}^\top\Lambda\right) \\
&= \sum_{i=1}^{N} -\frac{1}{2}\boldsymbol{h}_i^\top \Lambda \boldsymbol{h}_i + \frac{1}{2}\sum_{i=1}^{N}\boldsymbol{h}_i^\top\Lambda\boldsymbol{h}_i = 0.
\end{aligned}$$

□

Proposition E.2.2. *Let $J(\boldsymbol{x}) : \mathbb{R}^N \to \mathbb{R}$ be a smooth function, for all $x \in \mathbb{R}^N$ and for $c \in \mathbb{R}_{>0}$, define $\bar{J}(\boldsymbol{x}) := J(\boldsymbol{x}) + \frac{c}{2} J(\boldsymbol{x})^2$. Assuming $(1 + cJ(\boldsymbol{x})) \neq 0$, then $\boldsymbol{x}^\star$ is the stationary point of $\bar{J}(\boldsymbol{x})$ iff $\boldsymbol{x}^\star$ is the stationary point for $J(\boldsymbol{x})$.*

Proof. Let $\boldsymbol{x}^\star$ be a stationary point of $J(\boldsymbol{x})$, meaning that $\nabla J(\boldsymbol{x}^\star) = 0$. The stationary points for $\bar{J}(\boldsymbol{x})$ can be obtained from:

$$\frac{d\bar{J}}{dx} = (\nabla J(\boldsymbol{x}) + cJ(\boldsymbol{x})\nabla J(\boldsymbol{x})) = (1 + cJ(\boldsymbol{x}))\, \nabla J(\boldsymbol{x}). \tag{E.2.1}$$

It is easy to see from (E.2.2) that if $\boldsymbol{x} = \boldsymbol{x}^*$, $\nabla J(\boldsymbol{x}^*) = 0$, we have that $\frac{d\bar{J}}{dx}\Big|_{x^*} = 0$, meaning that all the stationary points of $J(\boldsymbol{x})$ are stationary points of $\bar{J}(\boldsymbol{x})$.

To show the other way, let $\boldsymbol{x}^\star$ be stationary point of $\bar{J}(\boldsymbol{x})$ i.e. $\nabla \bar{J}(\boldsymbol{x}^\star) = 0$. Assuming $(1 + cJ(\boldsymbol{x}^\star)) \neq 0$, then from (E.2.1) for all $c \in \mathbb{R}_{>0}$, we have

$$(1 + cJ(\boldsymbol{x}^\star))\, \nabla J(\boldsymbol{x}^\star) = 0,$$

implying that $\nabla J(\boldsymbol{x}^\star) = 0$. □

Based on the above propositions, we stabilize our original objective function (5.2.1) to keep it bounded and hence is suitable for minimization with Gradient-descent methods. Without the reconstruction errors, the stabilized objective function is

$$\min_{U,V,\boldsymbol{h}_i} \mathcal{J}_t + \frac{c}{2}\mathcal{J}_t^2.$$

Denoting $\bar{J} = \mathcal{J}_t + \frac{c_{stab}}{2}\mathcal{J}_t^2$. Since the derivatives of $\mathcal{J}_t$ are given by (5.2.2), the stationary points of $\bar{J}$ are:

$$\begin{cases} \frac{\partial \bar{J}}{\partial V} = (1 + c_{stab}\mathcal{J}_t)\left(-\sum_{i=1}^N \phi_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top + \eta_1 V\right) = 0, \\ \frac{\partial \bar{J}}{\partial U} = (1 + c_{stab}\mathcal{J}_t)\left(-\sum_{i=1}^N \phi_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top + \eta_2 U\right) = 0, \\ \frac{\partial \bar{J}}{\partial \boldsymbol{h}_i} = (1 + c_{stab}\mathcal{J}_t)\left(-V^\top \phi_1(\boldsymbol{x}_i) - U^\top \phi_2(\boldsymbol{y}_i) + \lambda \boldsymbol{h}_i\right) = 0, \end{cases}$$

which gives the following solution:

$$\begin{cases} V = \frac{1}{\eta_1}\sum_{i=1}^N \phi_1(\boldsymbol{x}_i)\boldsymbol{h}_i^\top, \\ U = \frac{1}{\eta_2}\sum_{i=1}^N \phi_2(\boldsymbol{y}_i)\boldsymbol{h}_i^\top, \\ \lambda \boldsymbol{h}_i = V^\top \phi_1(\boldsymbol{x}_i) + U^\top \phi_2(\boldsymbol{y}_i), \end{cases}$$

assuming $1 + c_{stab}\mathcal{J}_t \neq 0$. Elimination of V and U yields the following eigenvalue problem $\left[\frac{1}{\eta_1}K_1 + \frac{1}{\eta_2}K_2\right]\boldsymbol{h}^\top = \boldsymbol{h}^\top \Lambda$, which is indeed the same solution for $c_{stab} = 0$ in (5.2.1) and (5.2.3).

E.3 Architecture details

See Table E.3.1 and E.3.2 for details on model architectures, datasets, and hyperparameters used in Section 5.2, and double precision is used for training the Gen-RKM model. The PyTorch library in Python was used as the programming language with an 8GB NVIDIA QUADRO P4000 GPU.

Table E.3.1: Datasets and hyperparameters used for the experiments. The bandwidth of the Gaussian kernel for generation corresponds to the bandwidth that gave the best performance determined by cross-validation on the MNIST classification problem.

Dataset	N	d	N_{subset}	s	m	σ	n_r	l
MNIST	60000	28×28	10000	500	50	1.3	4	10
Fashion-MNIST	60000	28×28	500	100	5	/	/	10
CIFAR-10	60000	$32 \times 32 \times 3$	500	500	5	/	/	10
CelebA	202599	$128 \times 128 \times 3$	3000	15	5	/	/	20
Dsprites	737280	64×64	1024	2/10	5	/	/	/
Teapot	200000	$64 \times 64 \times 3$	1000	5/10	100	/	/	/
Sketchy	75471	$64 \times 64 \times 3$	1000	30	100	/	/	/

Table E.3.2: Details of model architectures used in Section 5.2. All convolutions and transposed-convolutions are with stride 2 and padding 1. Unless stated otherwise, the layers have Parametric-RELU ($\alpha = 0.2$) activation function, except the output layers of the pre-image maps which have a sigmoid activation function.

Dataset	**Optimizer** (Adam)		**Architecture** $\mathcal{X}$	$\mathcal{Y}$
MNIST/ fMNIST	1e-3	Input	28x28x1	10 (One-hot encoding)
		Feature-map (fm)	Conv 32x4x4; Conv 64x4x4; FC 128 (Linear)	FC 15, 20 (Linear)
		Pre-image map	reverse of fm	reverse of fm
		Latent space dim.		500/100
CIFAR-10	1e-3	Input	32x32x3	10 (One-hot encoding)
		Feature-map (fm)	Conv 64x4x4; Conv 128x4x4; FC 128 (Linear)	FC 15, 20
		Pre-image map	reverse of fm	reverse of fm
		Latent space dim.		500
CelebA	1e-4	Input	64x64x3	-
		Feature-map (fm)	Conv 32x4x4; Conv 64x4x4; Conv 128x4x4; Conv 256x4x4 ; FC 128 (Linear)	-
		Pre-image map	reverse of fm	-
		Latent space dim.		15
Dsprites	1e-4	Input	64x64x1	-
		Feature-map (fm)	Conv 20x4x4; Conv 40x4x4; Conv 80x4x4; FC 128 (Linear)	-
		Pre-image map	reverse of fm	-
		Latent space dim.		2/10
Teapot	1e-4	Input	64x64x3	-
		Feature-map (fm)	Conv 30x4x4; Conv 60x4x4; Conv 90x4x4; FC 128 (Linear)	-
		Pre-image map	reverse of fm	-
		Latent space dim.		5/10
Sketchy	1e-4	Input	64x64x3	-
		Feature-map (fm)	Conv 40x4x4; Conv 80x4x4; Conv 160x4x4; FC 128 (Linear)	-
		Pre-image map	reverse of fm	-
		Latent space dim.		30

E.4 Extra experiments and visualizations

E.4.1 Bilinear interpolation

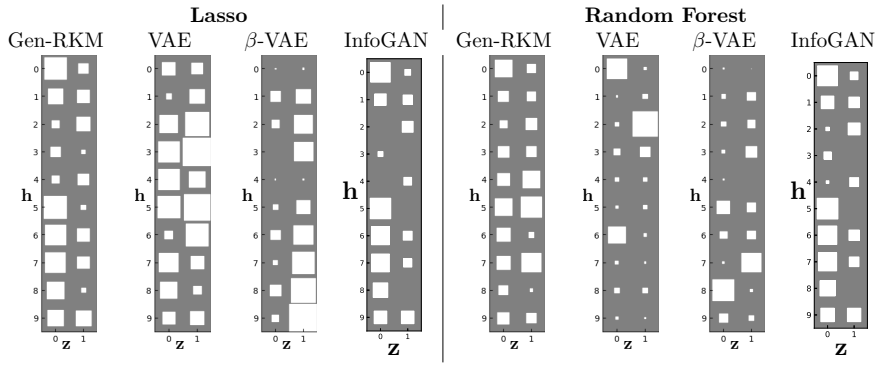
Given four vectors $\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3$ and $\boldsymbol{h}_4$ (reconstructed images from these vectors are shown at the corners of Figures 5.2.3e, 5.2.3f), the interpolated vector $\boldsymbol{h}^\star$ is given by:

$$\boldsymbol{h}^\star = (1-\alpha)(1-\gamma)\boldsymbol{h}_1 + \alpha(1-\gamma)\boldsymbol{h}_2 + \gamma(1-\alpha)\boldsymbol{h}_3 + \gamma\alpha\boldsymbol{h}_4,$$

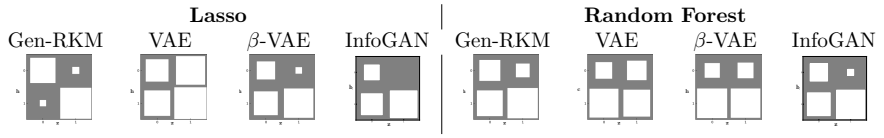
with $0 \leq \alpha, \gamma \leq 1$. This $\boldsymbol{h}^\star$ is then used in step 8 of the generation procedure of Gen-RKM algorithm (see Algorithm 5.2.1) to compute $\boldsymbol{x}^\star$.

E.4.2 Visualizing the disentanglement metric

In this section, we show the Hinton plots to visualize the disentanglement scores as shown in Table 5.2.2. Following the conventions of [72], $\boldsymbol{z}$ represents the ground-truth data generating factors. Figures E.4.1 and E.4.2 shows the Hinton plots on DSprites and Teapot datasets using Lasso and Random Forest regressors for various algorithms. Here the white square size indicates the magnitude of the *relative importance* of the latent code $\boldsymbol{h}_i$ in predicting $\boldsymbol{z}_i$.

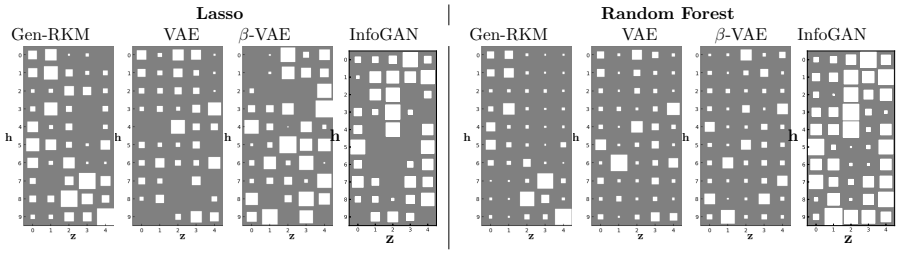


(a) $h_{dim} = 10$

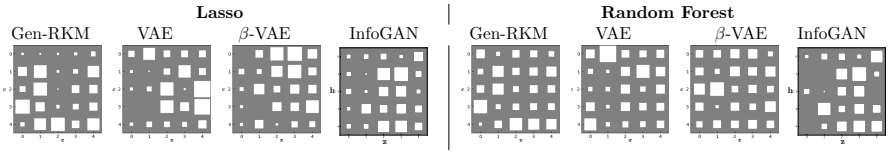


(b) $h_{dim} = 2$

Figure E.4.1: Relative importance matrix as computed by Lasso and Random Forest regressors on DSprites dataset for $h_{dim} \in \{10, 2\}$ against the underlying data generating factors $z_{dim} = 2$ corresponding to x, y positions of object.



(a) $h_{dim} = 10$



(b) $h_{dim} = 5$

Figure E.4.2: Relative importance matrix as computed by Lasso and Random Forest regressors on Teapot dataset for $h_{dim} \in \{10, 5\}$ against the underlying data generating factors $z_{dim} = 5$ corresponding to azimuth, elevation and colors red, green and blue of the teapot object.

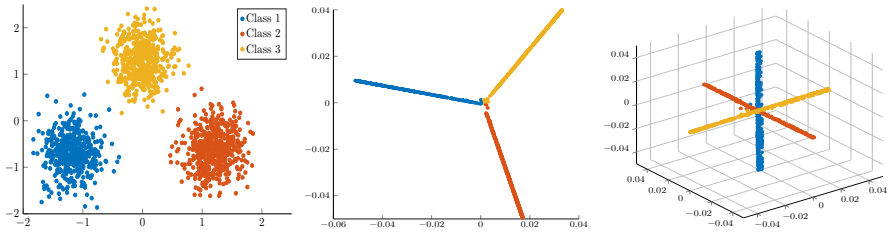
E.4.3 Further empirical results

Table E.4.1: Training time per epoch comparisons (in seconds with standard deviation over 10 epochs) on MNIST and CelebA datasets. The architecture is the same as shown in Table E.3.2 below with mini-batch size 100 and batch-size 2000. In both cases Info-GAN is the most computationally expensive due to the additional auxiliary network and two backward passes per iteration. β-TCVAE has the second-worst computation times due to relatively more complicated ELBO objective. VAE is marginally better for MNIST whereas the Gen-RKM outperforms for CelebA. This could be due to a significantly large number of parameters for CelebA architecture which increases the computational burden of VAE. However, due to the fixed computational cost of eigendecomposition (for fixed mini-batch size), the latent variables in Gen-RKM are computed with this fixed cost.

Dataset	Gen-RKM	VAE	Info-GAN	β-TCVAE
MNIST	0.275 ($\pm$0.042)	**0.223** ($\pm$0.013)	0.372 ($\pm$0.044)	0.318 ($\pm$0.031)
CelebA	**4.274** ($\pm$0.147)	4.308 ($\pm$0.112)	5.815 ($\pm$0.131)	5.201 ($\pm$0.152)

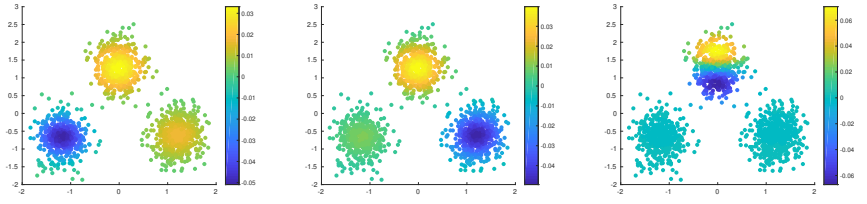
E.4.4 Illustration on toy example using a Gaussian kernel

Here we demonstrate the application of Gen-RKM / Kernel PCA using a Gaussian kernel with $\sigma = 0.5$ on a 3 mode Gaussian dataset. The dataset is shown in Figure E.4.3 together with the first 5 Principal Components (PCs) of the latent space. The method looks for PCs that explain the most variance. One can see that moving along the first component in latent space corresponds to changing classes $3 \rightarrow 2 \rightarrow 1$, whereas moving along the second component corresponds to changing classes $2 \rightarrow 3 \rightarrow 1$. For PCs 3 to 5, the model shows disentanglement of the 3 classes, i.e. each Gaussian cluster is mapped to a specific component. Moving along one of these components only changes the *within class* variation. This behavior is further confirmed by the experiment in Figure E.4.4. Here we visualize again the dataset where now the color corresponds to the value of the datapoint in latent space.

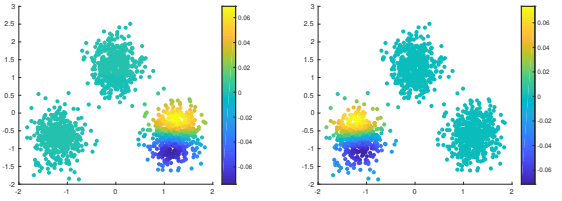


(a) Input data (b) PCs 1 & 2 (c) PCs 3,4 & 5

Figure E.4.3: Visualization of the toy dataset together with the first 5 Principal Components (PCs) of the latent space of the Gen-RKM model.



(a) Latent dim. 1 (b) Latent dim. 2 (c) Latent dim. 3



(d) Latent dim. 4 (e) Latent dim. 5

Figure E.4.4: Visualization of the traversals along the Principal components. Here the color corresponds to the value of the datapoint in latent space.



Figure E.4.5: Comparing Gen-RKM and standard VAE for reconstruction and generation quality. In reconstruction MNIST and reconstruction CelebA, uneven columns correspond to the original image, even columns to the reconstructed image. The faces in the uneven columns are real, The faces in the even columns are fake.

Appendix F

Disentangled representation learning and generation with manifold optimization

This appendix contains the supplementary material of the work presented in Section 5.3. Section F.1 contains the details of the theoretical result. Next, datasets, hyperparameters and algorithm are described in Section F.2. Extra ablation studies are listed in Section F.3. Finally, Sections F.4 and F.5 give additional results and figures respectively.

F.1 Theoretical results

F.1.1 Proof of Lemma 5.3.1

We first quote a result that is used in the context of optimization ([168], Lemma 1.2.4). Let f a function with L_a-Lipschitz continuous Hessian. Then, it holds

$$\Big| \underbrace{f(\boldsymbol{y}_1) - f(\boldsymbol{y}) - \nabla f(\boldsymbol{y})^\top (\boldsymbol{y}_1 - \boldsymbol{y}) - \frac{1}{2}(\boldsymbol{y}_1 - \boldsymbol{y})^\top \operatorname{Hess}_{\boldsymbol{y}}[f](\boldsymbol{y}_1 - \boldsymbol{y})}_{r(\boldsymbol{y}_1 - \boldsymbol{y})} \Big| \leq \tfrac{L_a}{6} \|\boldsymbol{y}_1 - \boldsymbol{y}\|_2^3. \tag{F.1.1}$$

Then, we calculate the power series expansion of $f(\boldsymbol{y}) = [\boldsymbol{x} - \psi(\boldsymbol{y})]_a^2$ and take the expectation with respect to $\epsilon \sim \mathcal{N}(0, \mathbb{I})$. Firstly, we have $\nabla f(\boldsymbol{y}) =$

$-2[\boldsymbol{x} - \psi(\boldsymbol{y})]_a \nabla\psi_a(\boldsymbol{y})$ and

$$\mathrm{Hess}_{\boldsymbol{y}}[f] = 2\nabla\psi_a(\boldsymbol{y})\nabla\psi_a(\boldsymbol{y})^\top - 2[\boldsymbol{x} - \psi(\boldsymbol{y})]_a \,\mathrm{Hess}_{\boldsymbol{y}}[\psi_a].$$

Then, we use (F.1.1) with $\boldsymbol{y}_1 - \boldsymbol{y} = \sigma U\epsilon$. By taking the expectation over ϵ, notice that the order 1 term in σ vanishes since $\mathbb{E}_\epsilon[\epsilon] = 0$. We find

$$\begin{aligned}\mathbb{E}_\epsilon[\boldsymbol{x} - \psi(\boldsymbol{y} + \sigma U\epsilon)]_a^2 =& [\boldsymbol{x} - \psi(\boldsymbol{y})]_a^2 + \sigma^2 \,\mathrm{Tr}\left(U^\top \nabla\psi_a(\boldsymbol{y})\nabla\psi_a(\boldsymbol{y})^\top U\right) \\ & -\sigma^2[\boldsymbol{x} - \psi(\boldsymbol{y})]_a \,\mathrm{Tr}\left(U^\top \mathrm{Hess}_y[\psi_a]U\right) + \mathbb{E}_\epsilon r(\sigma U\epsilon),\end{aligned}$$

where we used that $\mathbb{E}_\epsilon[\epsilon^\top M\epsilon] = \mathrm{Tr}[M]$ for any symmetric matrix M since $\mathbb{E}_\epsilon[\epsilon_i\epsilon_j] = \delta_{ij}$. Next, denote $R_a(\sigma) = \mathbb{E}_\epsilon r(\sigma U\epsilon)$ we can use the Jensen inequality and subsequently (F.1.1)

$$|R_a(\sigma)| = |\mathbb{E}_\epsilon r(\sigma U\epsilon)| \leq \mathbb{E}_\epsilon |r(\sigma U\epsilon)| \leq \frac{L_a}{6}\mathbb{E}_\epsilon \|\sigma U\epsilon\|_2^3.$$

Next, we notice that $\|\sigma U\epsilon\|_2 = \sigma(\epsilon^\top U^\top U\epsilon)^{1/2} = \sigma\|\epsilon\|_2$. It is useful to notice that $\|\epsilon\|_2$ is distributed according to a chi distribution. By using this remark, we find

$$|R_a(\sigma)| \leq \sigma^3 \frac{L_a}{6}\mathbb{E}_\epsilon\|\epsilon\|_2^3 = \sigma^3 \frac{L_a}{6}\frac{\sqrt{2}(m+1)\Gamma((m+1)/2)}{\Gamma(m/2)},$$

where the last equality uses the expression for the third moment of the chi distribution and where the Gamma function Γ is the extension of the factorial to the complex numbers.

F.1.2 Details on Evidence Lower Bound for St-KM model

We discuss here the details of the ELBO. The first term in (5.3.5) is

$$\begin{aligned}\mathbb{E}_{q_U(\boldsymbol{z}|\boldsymbol{x}_i)}[\log(p(\boldsymbol{x}_i|\boldsymbol{z}))] = & -\frac{1}{2\sigma_0^2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbb{I})}\|\boldsymbol{x}_i - \psi_{\boldsymbol{\xi}}(\mathbb{P}_U\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \sigma\mathbb{P}_U\epsilon + \delta\mathbb{P}_{U^\perp}\epsilon)\|_2^2 \\ & - \frac{d}{2}\log(2\pi\sigma_0^2).\end{aligned}$$

Where we used the following reparameterization, following [122]: $\mathbb{E}_{q_U(\boldsymbol{z}|\boldsymbol{x}_i)}[f(\mathbf{z})] = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbb{I})}\left[f\left(\mathbb{P}_U\phi_{\boldsymbol{\theta}}(\boldsymbol{x}) + (\sigma\mathbb{P}_U + \delta\mathbb{P}_{U^\perp})\epsilon\right)\right]$, with $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\psi_{\boldsymbol{\xi}}(\boldsymbol{z}), \sigma_0^2\mathbb{I})$ and $q_U(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mathbb{P}_U\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2\mathbb{P}_U + \delta^2\mathbb{P}_{U^\perp})$. Clearly, the above expectation can be written as follows

$$\mathbb{E}_\epsilon\mathbb{E}_{\epsilon_\perp}\|\boldsymbol{x}_i - \psi_{\boldsymbol{\xi}}(\mathbb{P}_U\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \sigma U\epsilon + \delta U_\perp\epsilon_\perp)\|_2^2,$$

with $\epsilon \sim \mathcal{N}(0, \mathbb{I}_m)$ and $\epsilon_\perp \sim \mathcal{N}(0, \mathbb{I}_{\ell-m})$. Hence, we fix $\sigma_0^2 = 1/2$ and take $\delta > 0$ to a numerically small value. For the other terms of (5.3.5), we use the formula giving the KL divergence between multivariate normals. Let $\mathcal{N}_0$ and $\mathcal{N}_1$ be ℓ-variate normal distributions with mean $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$ and covariance Σ_0, Σ_1 respectively. Then,

$$\mathrm{KL}(\mathcal{N}_0, \mathcal{N}_1) = \frac{1}{2}\Big\{ \mathrm{Tr}(\Sigma_1^{-1}\Sigma_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \Sigma_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - \ell + \log\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right) \Big\}$$

By using this identity, we find the second term of (5.3.5):

$$\mathrm{KL}[q_U(\boldsymbol{z}|\boldsymbol{x}_i), q(\boldsymbol{z}|\boldsymbol{x}_i)] = \frac{1}{2}\Big\{ \frac{m\sigma^2 + (\ell - m)\delta^2}{\gamma^2} + \frac{1}{\gamma^2}\|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \mathbb{P}_U \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2$$

$$- \ell + \log\big(\frac{\gamma^{2\ell}}{\sigma^{2m}\delta^{2(\ell-m)}}\big)\Big\},$$

where $q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), \gamma^2 \mathbb{I}_\ell)$. For the third term in (5.3.5), we find

$$\mathrm{KL}[q_U(\boldsymbol{z}|\boldsymbol{x}_i), p(\boldsymbol{z})] = \frac{1}{2}\Big\{ \mathrm{Tr}((\sigma^2 \mathbb{P}_U + \delta^2 \mathbb{P}_{U^\perp})\Sigma^{-1})$$

$$+ (\mathbb{P}_U \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^\top \Sigma^{-1} (\mathbb{P}_U \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \log\det(\Sigma) - \ell - \log(\sigma^{2m}\delta^{2(\ell-m)})\Big\},$$

with $p(\boldsymbol{z}) = \mathcal{N}(0, \Sigma)$. By averaging over $i = 1, \ldots, n$, we obtain

$$\frac{1}{n}\sum_{i=1}^n \mathrm{KL}[q_U(\boldsymbol{z}|\boldsymbol{x}_i), p(\boldsymbol{z})] = \frac{1}{2}\Big\{ \mathrm{Tr}((\sigma^2 \mathbb{P}_U + \delta^2 \mathbb{P}_{U^\perp})\Sigma^{-1})$$

$$+ \mathrm{Tr}(\mathbb{P}_U C_{\boldsymbol{\theta}} \mathbb{P}_U \Sigma^{-1}) + \log\det(\Sigma) - \ell - \log(\sigma^{2m}\delta^{2(\ell-m)})\Big\},$$

where we used the cyclic property of the trace and $C_{\boldsymbol{\theta}} = \frac{1}{n}\sum_{i=1}^n \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)^\top$. Finally, the estimation of the optimal Σ can be done in parallel to the Maximum Likelihood Estimation of the covariance matrix of a multivariate normal.

F.2 Datasets, hyperparameters and algorithm

We refer to Table F.2.1 and Table F.2.2 for more details on model architectures, datasets and hyperparameters used in Section 5.3. All models were trained on full-datasets and for maximum 1000 epochs. Further, all datasets are scaled between [0-1] and are resized to 28×28 dimensions except Dsprites and Cars3D. The PyTorch library (single precision) in Python was used as the programming language on 8GB NVIDIA QUADRO P4000 GPU. See Algorithm 5.3.1 for training the St-RKM model. In the case of FactorVAE, the discriminator architecture is the same as proposed in the original paper [120].

Table F.2.1: Datasets and hyperparameters used for the experiments. N is the number of training samples, d the input dimension (resized images), m the subspace dimension and M the minibatch size.

Dataset	N	d	m	M
MNIST	60000	28×28	10	256
fMNIST	60000	28×28	10	256
SVHN	73257	$32 \times 32 \times 3$	10	256
Dsprites	737280	64×64	5	256
3Dshapes	480000	$64 \times 64 \times 3$	6	256
Cars3D	17664	$64 \times 64 \times 3$	3	256

Table F.2.2: Model architectures. All convolutions and transposed-convolutions are with stride 2 and padding 1. Unless stated otherwise, layers have Parametric-RELU ($\alpha = 0.2$) activation functions, except output layers of the pre-image maps which have sigmoid activation functions (since input data is normalized $[0, 1]$). Adam and CayleyAdam optimizers have learning rates 2×10^{-4} and 10^{-4} respectively. Pre-image map/decoder network is always taken as transposed of feature-map/encoder network. $c = 48$ for Cars3D; and $c = 64$ for all others. Further, $\hat{k} = 3$ and stride 1 for MNIST, fMNIST, SVHN and 3Dshapes; and $\hat{k} = 4$ for others. SVHN and 3Dshapes are resized to 28×28 input dimensions.

Dataset	**Architecture**
MNIST/fMNIST/ /SVHN/3Dshapes/ Dsprites/Cars3D	$\boldsymbol{\phi_\theta}(\cdot) = \begin{cases} Conv\ [c] \times 4 \times 4; \\ Conv\ [c \times 2] \times 4 \times 4; \\ Conv\ [c \times 4] \times \hat{k} \times \hat{k}; \\ FC\ 256; \\ FC\ 50\ (Linear) \end{cases}$ $\boldsymbol{\psi_\varsigma}(\cdot) = \begin{cases} FC\ 256; \\ FC\ [c \times 4] \times \hat{k} \times \hat{k}; \\ Conv\ [c \times 2] \times 4 \times 4; \\ Conv\ [c] \times 4 \times 4; \\ Conv\ [c]\ (Sigmoid) \end{cases}$

F.3 Ablation studies

Analysis of the impact of the KPCA loss: In this section, we show an ablation study on the KPCA loss. We repeat the experiments of Section 5.3.5 on the 3DShapes dataset, where we consider 3 different variants of the proposed model:

1. **St-RKM:** The KPCA loss is optimized in a stochastic manner using the Cayley Adam optimizer, as proposed in Section 5.3.
2. **Gen-RKM:** The KPCA loss is optimized exactly at each step using an eigendecomposition (this corresponds to the algorithm in [175]).

3. **Post-RKM:** The PCA is performed at the end as a post-processing step. During training, only the reconstruction loss is minimized (standard AE).

The encoder/decoder maps are the same across all the models, and for the Post-RKM model, additional linear layers are used to map the latent space with the subspace. From Table F.3.1, we conclude that optimizing the KPCA loss during training improves disentanglement. Moreover, using a stochastic algorithm improves computation time and scalability with only a slight decrease in disentanglement score. Note that calculating the exact eigendecomposition at each step (Gen-RKM) comes with some numerical difficulties. In particular, double floating-point precision has to be used together with careful tuning of the number of principal components to avoid numerical instabilities when calculating gradients through the eigendecomposition. These problems are resolved when using the St-RKM training algorithm.

		St-**RKM**	**Gen-RKM**	**Post-RKM**
Train. time		3.01 (±0.71)	9.21 (±0.54)	**2.87** (±0.33)
Disent. score	Lasso	0.40 (±0.02)	**0.44** (±0.01)	0.35 (±0.01)
	RF	0.27 (±0.01)	**0.31** (±0.02)	0.22 (±0.02)

Table F.3.1: Training timings per epoch (in minutes) and disentanglement scores ([98]) for different variants of RKM when trained on the 3Dshapes dataset. Gen-RKM has the highest training time since it requires the eigendecomposition of the kernel matrix at every iteration. This step is approximated by the St-RKM model which involves stochastic optimization on the Stiefel manifold modeled as the eigenvectors of the covariance matrix. Finally, for the Post-RKM, PCA in the post-processing step speeds up training but severely degrades the disentanglement. This is because the basis of the latent space has now changed and hence the decoder cannot reconstruct the training data.

	St-**RKM**	**VAE**	β-**VAE**	**FactorVAE**	**InfoGAN**
MNIST	24.63 (±0.22)	36.11 (±1.01)	42.81 (±2.01)	35.48 (±0.07)	45.74 (±2.93)
fMNIST	61.44 (±1.02)	73.47 (±0.73)	75.21 (±1.11)	69.73 (±1.54)	84.11 (±2.58)

Table F.3.2: FID scores (lower is better, with standard deviations) computed on randomly generated 8000 images when trained with architecture and hyperparameters adapted from [70].

Smaller encoder/decoder architecture: In this section, we analyze the impact of the encoder/decoder architecture on the generation quality of considered models. The generation quality experiment of Section 5.3.5 is

repeated on the fMNIST and MNIST dataset, where the architecture and hyperparameters are adapted from [70]. From Table F.3.2 and Figure F.3.1, we see that the overall FID scores and generation quality have improved, however, the relative scores among the models did not change significantly.

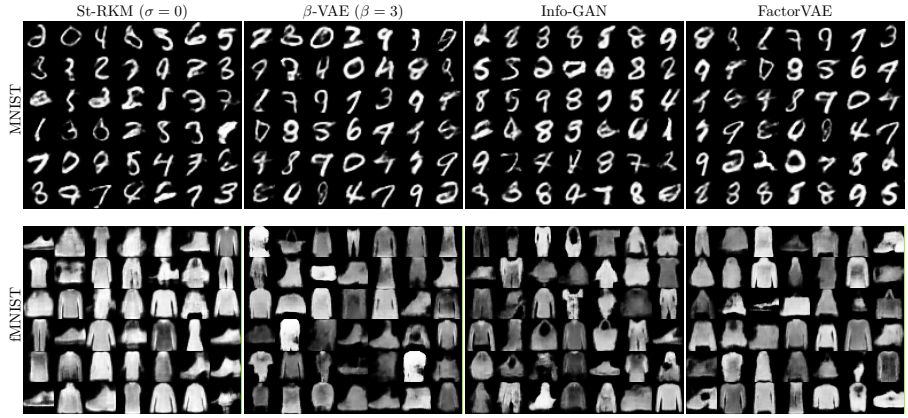


Figure F.3.1: Samples of randomly generated images used to compute FID scores; see Table F.3.2.

Analysis of St-RKM with a fixed U : We discuss here the role of the optimization of $\mathrm{St}(\ell, m)$ on disentanglement in the case of a classical AE loss ($\sigma = 0$). To do so, a matrix $\tilde{U} \in \mathrm{St}(\ell, m)$ is generated randomly[1] and kept fixed during the training of the following optimization problem

$$\min_{\boldsymbol{\theta},\boldsymbol{\xi}} \lambda \frac{1}{n} \sum_{i=1}^{n} L^{(0)}_{\boldsymbol{\xi},\tilde{U}} \left(\boldsymbol{x}_i, \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) + \underbrace{\frac{1}{n} \sum_{i=1}^{n} \|\mathbb{P}^{(\varepsilon)}_{\tilde{U}^\perp} \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2}_{\text{regularized PCA objective}}, \tag{F.3.1}$$

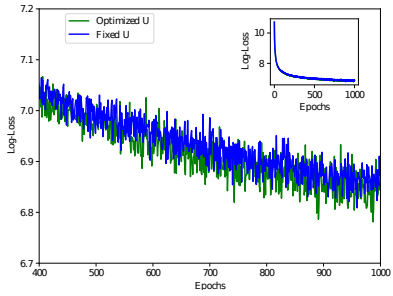
with $\lambda = 1$ and where $\varepsilon \geq 0$ is a regularization constant and where the regularized (or mollified) projector $\mathbb{P}^{(\varepsilon)}_{\tilde{U}^\perp} = \varepsilon(\tilde{U}\tilde{U}^\top + \varepsilon \mathbb{I}_\ell)^{-1}$ is used in order to prevent numerical instabilities. Indeed, if $\varepsilon = 0$, the second term in (F.3.1) (PCA term) is not strictly convex as a function of $\phi_{\boldsymbol{\theta}}$, since this quadratic form has flat directions along the column subspace of $\tilde{U}$. Our numerical simulations in single precision pytorch with $\varepsilon = 0$ exhibit instabilities, i.e., the PCA term in (F.3.1) takes negative values during the training. Hence, the regularized projector is introduced so that the PCA quadratic is strongly convex for $\varepsilon > 0$. This instability is not observed in the training of (St-RKM) where U is not

[1] Using a random $\tilde{U} \in \mathrm{St}(\ell, m)$ can be interpreted as sketching the encoder map in the spirit of Randomized Orthogonal Systems (ROS) sketches (see [252]).

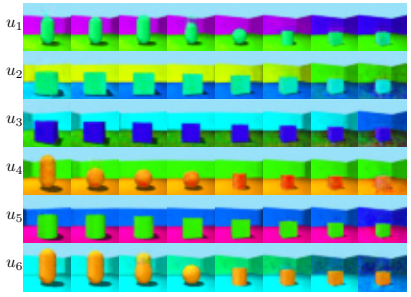
fixed. This is one asset of our training procedure using optimization over Stiefel manifold. Explicitly, the regularized projector satisfies the following properties

- $\mathbb{P}^{(\varepsilon)}_{\tilde{U}^\perp}\boldsymbol{u}_\perp = \boldsymbol{u}_\perp$ for all $\boldsymbol{u}_\perp \in (\text{range}(U))^\perp$,
- $\mathbb{P}^{(\varepsilon)}_{\tilde{U}^\perp}\boldsymbol{u} = \varepsilon\boldsymbol{u}$ for all $\boldsymbol{u} \in \text{range}(U)$.

Clearly, thanks to the push-through identity, we have the alternative expression $\mathbb{P}^{(\varepsilon)}_{\tilde{U}^\perp} = \mathbb{I} - U(U^\top U + \varepsilon\mathbb{I}_m)^{-1}U^\top$. Therefore, it holds $\lim_{\varepsilon\to 0}\mathbb{P}^{(\varepsilon)}_{\tilde{U}^\perp} = \mathbb{P}_{\tilde{U}^\perp}$, as it should. In our experiments, we set $\varepsilon = 10^{-5}$. If $\varepsilon \leq 10^{-6}$, the regularized PCA objective in (F.3.1) takes negative values after a few epochs due to the numerical instability as mentioned above.



(a) Loss-evolution



(b) Latent space Traversal

Figure F.3.2: (a) Loss evolution (log plot) during the training of (F.3.1) over 1000 epochs with $\varepsilon = 10^{-5}$ once with Cayley-Adam optimizer (green curve) and then without (blue curve). (b) Traversals along the principal components when the model was trained with a fixed U, i.e. with the objective given by (F.3.1) and $\varepsilon = 10^{-5}$. There is no clear isolation of a feature along any of the principal components indicating further that optimizing over U is key for better disentanglement.

In Figure F.3.2a, the evolution of the training objective (F.3.1) is displayed. It can be seen that the final objective has a lower value [$\exp(6.78) \approx 881$] when U is optimized compared to its fixed counterpart [$\exp(6.81) \approx 905$] showing the merit of optimizing over Stiefel manifold for the same parameter ε. Hence the subspace determined by range(U) has to be adapted to the encoder and decoder networks. In other words, the training over $\boldsymbol{\theta}, \boldsymbol{\xi}$ is not sufficient to minimize the $\text{St}(\ell, m)$ objective with Adam. Figure F.3.2b further explores the latent traversals in the context of this ablation study. In the top row of Figure F.3.2b (latent traversal in the direction of u_1), both the shape of the object and the

wall hue are changing. A coupling between wall hue and shape is also visible in the bottom row of this figure.

F.4 Additional empirical results

Figure F.4.1 reports the training curves of the models considered in Section 5.3. This confirms that the models have been trained with a sufficiently large number of epochs. In Figure F.4.2, a two-dimensional projection of the embedding of the Dsprites data set in latent space is displayed. A successful optimization procedure is expected to yield a Gaussian cluster in feature space. Since random generation requires sampling from a latent normal distribution, the models which do not respect this constraint strongly are expected to generate poorer quality images (FID/SWD scores).

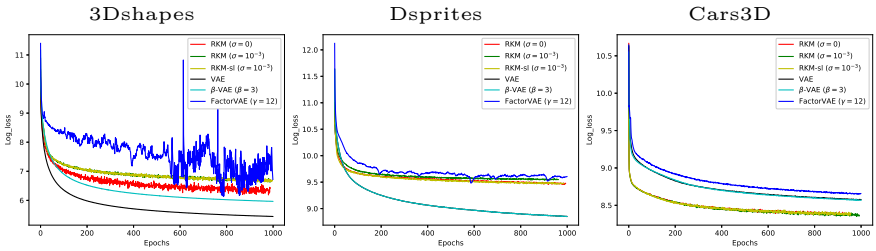


Figure F.4.1: Reconstruction errors (log-scale) of various models during the training process over 1000 epochs, after which no significant improvement of the objectives can be observed. Further, the final losses may not be compared in an absolute sense since the objective functions of the models are different.

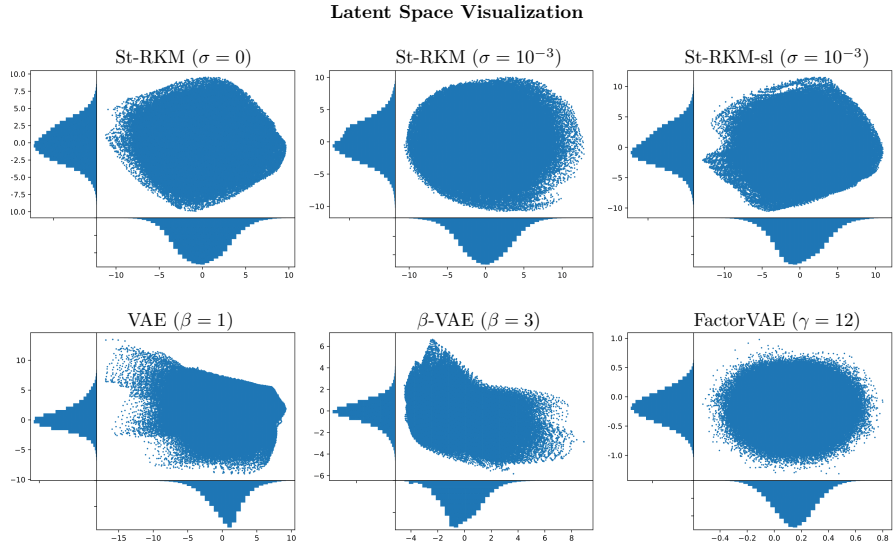


Figure F.4.2: Scatter plots of latent variables with histograms. The models were trained on full Dsprites dataset (737280 points). In the case of St-RKMs, the joint distributions are always centered around zero and elliptically distributed.

F.5 Additional figures

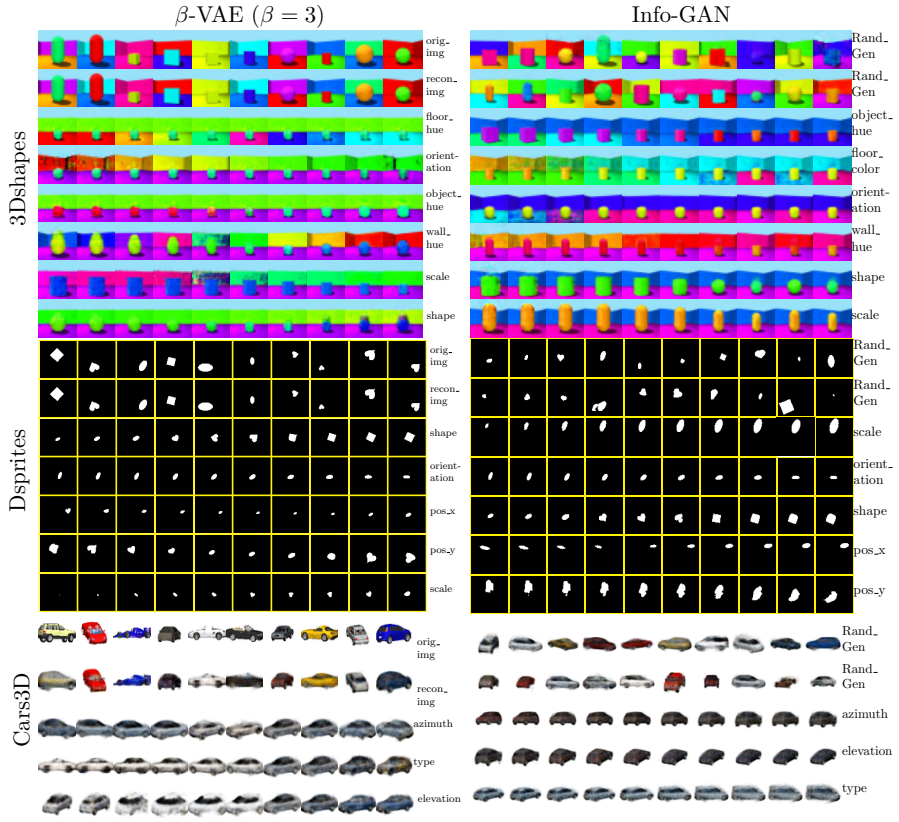


Figure F.5.1: β-VAE and Info-GAN: Traversals along principal components. First-two rows show the ground-truth and reconstructed images in the case of β-VAE, and randomly generated samples in the case of Info-GAN. Each subsequent row shows the generated images by traversing along a principal component in the latent space. The last column in each image shows the visually evaluated dominant factor of variation for reference purposes.

Appendix G

Outlier detection in non-elliptical data by kernel MRCD

This appendix contains the supplementary material of the work presented in Section 6.2. The algorithms of the initial estimators are described in Section G.1. Section G.2 gives the derivation for calculating the regularized covariance matrix based on the eigenvectors of the regularized kernel matrix. Section G.3 includes additional simulation results . Finally, Section G.4 contains additional figures for non-elliptical data.

G.1 Algorithms

Algorithms G.1.1 to G.1.4 below describe the initial estimators used. Algorithm G.1.5 carries out the subsequent refinement procedure.

Algorithm G.1.1 Spatial rank initial estimator

Input: kernel matrix K and the number h.
Initialize the vector with the n spatial ranks as $R = (0, \ldots, 0)^\top$.
Compute the spatial rank R_i of each observation using (6.2.10).
Define H as the set of h observations with the lowest values of R_i.
Determine the weights w_i using (6.2.7).
Return: the location weights w_i and the covariance weights $u_i := w_i$.

Algorithm G.1.2 Spatial sign covariance matrix initial estimator

Input: kernel matrix K.
Compute the spatial median $\boldsymbol{m}_{\mathcal{F}} = \sum_{i=1}^{N} \gamma_i \phi(\boldsymbol{x}_i)$ using Algorithm 2.
The location weights are taken as $w_i = \gamma_i$.
Compute the distances $||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_{\mathcal{F}}||$ to the spatial median from (6.2.6).
The covariance weights are then computed as $u_i = 1/||\phi(\boldsymbol{x}_i) - \boldsymbol{m}_{\mathcal{F}}||$.
Return: the location weights w_i and the covariance weights u_i .

Algorithm G.1.3 Spatial median initial estimator

Input: kernel matrix K and the number h.
Initialize the vector $\boldsymbol{\gamma} = (1/n, 1/n, \ldots, 1/n)^\top$ with length n.
For iteration = 1 to 10 do

1. Update the coefficients γ_i by

$$\gamma_{\text{new},i} := (K_{ii} - 2\gamma_{\text{old}}^T K_{.i} + \gamma_{\text{old}}^T K \gamma_{\text{old}})^{-\frac{1}{2}}$$

   where K_{ii} is the i-th diagonal element of K and $K_{.i}$ is the i-th column of K.

2. Normalize $\boldsymbol{\gamma}_{\text{new}} \leftarrow \boldsymbol{\gamma}_{\text{new}}/(\sum_{i=1}^{n} \gamma_{\text{new},i})$.

Compute the distance of each observation to the spatial median by (6.2.6).
Construct H as the set of the h observations with the lowest distances.
Determine the weights w_i using (6.2.7).
Return: the location weights w_i and the covariance weights $u_i := w_i$.

Algorithm G.1.4 Stahel-Donoho initial estimator

Input: kernel matrix K and the number h.
Initialize the vector of outlyingness values of the n observations as $\eta = (0, \ldots, 0)^\top$.
For $q = 1 : 500$ (number of directions) do

1. Select indices $i \neq j$ at random from $\{1, \ldots, n\}$
2. Initialize the vector $\lambda = (0, \ldots, 0)$ of length n and set $\boldsymbol{\lambda}(i) = +1$ and $\lambda(j) = -1$. This vector represents the direction in $\mathcal{F}$ formed by two observations.
3. Compute the projections of all n points on this direction as $a = K\boldsymbol{\lambda}/\sqrt{\boldsymbol{\lambda}^\top K \boldsymbol{\lambda}}$.
4. Compute the outlyingness of all n projected points as $r^{sd} = |a - \text{median}(a)|/\,\text{mad}(a)$, where mad(.) is the median absolution deviation.
5. Update the maximum outlyingness of each observation: $\eta_i^{\text{new}} = \max(\eta_i^{\text{old}}, r_i^d)$.

Define H as the set of h observations with the lowest η_i .
Determine the weights w_i using (6.2.7).
Return: the location weights w_i and the covariance weights $u_i := w_i$.

Algorithm G.1.5 Refinement of initial estimators

Input: kernel matrix K and number h, weights w_i to determine the initial estimate of location $\boldsymbol{c}_{\mathcal{F}}$ by (6.2.8) and weights u_i to compute the initial estimate of scatter $\hat{\Sigma}_{\mathcal{F}}$ by (6.2.9).
Project the data on the eigenvectors of $\hat{\Sigma}_{\mathcal{F}}$ using (6.2.11).
Compute the spatial median of the adjusted kernel matrix (6.2.12) using Algorithm 2, which gives the weights $\boldsymbol{\gamma}^*$.
Determine the distance $d^*_{\mathcal{F}}(\boldsymbol{x})$ of each observation by (6.2.13).
Define H as the set of h observations with the lowest distance.
Output: the refined subset H.

G.2 The special case of the linear kernel

Given an $n \times d$ matrix X of observations, its regularized covariance matrix is given by

$$\hat{\Sigma}_{\text{reg}} = (1-\rho)\hat{\Sigma}_{\mathcal{F}} + \rho\mathbb{I}_{\mathcal{F}} = \frac{1-\rho}{n-1}\tilde{\Phi}\tilde{\Phi}^\top + \rho\mathbb{I}_{\mathcal{F}} \ .$$

The regularized kernel matrix is defined as

$$\tilde{K}_{\text{reg}} = (1-\rho)\tilde{K} + (n-1)\rho\mathbb{I}_n = (1-\rho)\tilde{\Phi}^\top\tilde{\Phi} + (n-1)\rho\mathbb{I}_n.$$

For the linear kernel, the eigenvectors of the covariance matrix are given by

$$\boldsymbol{v}_{\mathcal{X}}^k = \sum_{i=1}^{n} (\boldsymbol{v}_{\mathcal{F}}^k)_i (\boldsymbol{x}_i - \boldsymbol{c}_{\mathcal{F}}),$$

where $\boldsymbol{c}_{\mathcal{F}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$ and $(\boldsymbol{v}_{\mathcal{F}}^k)_i$ is the i-th element of the k-th eigenvector of the centered kernel matrix. We can now write $\hat{\Sigma}_{\mathcal{F}}$ using the eigenvectors of the kernel matrix:

$$\hat{\Sigma}_{\mathcal{F}} = V^T\Lambda V = \frac{1}{n-1}\tilde{X}^\top\tilde{V}_{\mathcal{F}}\Lambda\tilde{V}_{\mathcal{F}}^\top\tilde{X},$$

where the $n \times n$ matrix $\tilde{V}_{\mathcal{F}} = \left[\boldsymbol{v}_1/\sqrt{\lambda_1}, \ldots, \boldsymbol{v}_n/\sqrt{\lambda_n}\right]$ contains the normalized eigenvectors of the centered kernel matrix and $\tilde{X}$ is the centered data matrix. The regularized covariance matrix is thus equal to

$$\hat{\Sigma}_{\text{reg}} = \frac{1-\rho}{n-1}\tilde{X}^\top\tilde{V}_{\mathcal{F}}\Lambda\tilde{V}_{\mathcal{F}}^\top\tilde{X} + \rho\mathbb{I}_n \ .$$

G.3 Additional experiments

G.3.1 Effect of the imposed condition number

The main text considers the condition number κ of a covariance matrix in two different places. The first occurrence is in the KMRCD algorithm itself. Indeed, the choice of the regularization parameter ρ in (6.2.4) must be such that the condition number $\kappa(\rho)$ given by (6.2.14) is at most 50. The second occurrence is in the simulation study with linear kernel, when in each replication a random correlation matrix Σ of type ALYZ is generated according to Section 4 of [8], who impose that such correlation matrices have condition number $\kappa(\text{ALYZ}) = 100$.

It is natural to wonder about the effect of both choices, and to what extent they interact. To study this we repeat part of the simulation of Section 6.2.5, for the

hardest settings with (n, d) set to $(400, 200)$ and $(200, 400)$ and with ε equal to 0 and 0.3 and $h = 0.5$. The contamination types are as before. We now allow the condition number κ(ALYZ) to be 100, 300 and 1000. The imposed condition number $\kappa(\rho)$ in the algorithm can now take the values 50, 100 and 1000. The averaged KL divergence over 100 replications is shown in Table G.3.1.

Table G.3.1: The KL divergence in function of the condition number of the ALYZ covariance matrix κ(ALYZ) and $\kappa(\rho)$ of the KMRCD algorithm with $h = 0.5$.

		Point contamination Value of κ(ALYZ)			Shift contamination Value of κ(ALYZ)			Cluster contamination Value of κ(ALYZ)		
		100	300	1000	100	300	1000	100	300	1000
	$\varepsilon = 0$:									
400×200	$\kappa(\rho) = 50$	127	146	197	128	147	198	127	145	197
	$\kappa(\rho) = 100$	158	166	196	159	166	197	159	167	194
	$\kappa(\rho) = 1000$	269	264	265	268	265	264	268	264	264
200×400	$\kappa(\rho) = 50$	674	761	969	677	759	969	675	759	943
	$\kappa(\rho) = 100$	1104	1168	1340	1108	1169	1326	1107	1169	1323
	$\kappa(\rho) = 1000$	6480	6380	6401	6492	6366	6367	6440	6375	6403
	$\varepsilon = 0.3$:									
400×200	$\kappa(\rho) = 50$	125	142	11536	125	143	205	125	144	251
	$\kappa(\rho) = 100$	156	163	20581	155	164	193	155	164	422
	$\kappa(\rho) = 1000$	265	10189	84460	256	185	209	257	189	209
200×400	$\kappa(\rho) = 50$	694	776	13729	692	786	1007	693	771	1041
	$\kappa(\rho) = 100$	1138	1199	25464	1157	1216	1416	1152	1213	1344
	$\kappa(\rho) = 1000$	6766	72332	195728	6906	1570	5405	6862	1568	5275

Comparing the choices of $\kappa(\rho)$, we observe that the strongest regularization ($\kappa(\rho) = 50$) consistently performed best (had the lowest KL), across all scenarios considered. That is, even when the true κ(ALYZ) is substantially above 50, setting $\kappa(\rho)$ to 50 did better or equally well than setting it to 100 or 1000. This provides some support for the choice $\kappa(\rho) = 50$ in the KMRCD algorithm.

G.3.2 Comparison of initial estimators

In this section we study the computation time and the performance of the four initial estimators used by KMRCD, in several scenarios:

1. as in the simulation with linear kernel (Section 6.2.5), with point contamination, $\varepsilon = 0.3$, $h = 0.5$, and $(n, d) = (400, 200)$;

2. as in the first setting, but now with $(n, d) = (200, 400)$ so there are more variables than cases;

3. the t-copula setting of the simulation with nonlinear kernels, as in Figure 6.2.4 of section 6.2.5, with $\varepsilon = 0.2$ and $h = 0.75$.

4. the circle-based setting of the simulation with nonlinear kernels, as in Figure 6.2.5, also with $\varepsilon = 0.2$ and $h = 0.75$.

Each setting is replicated 100 times, and in each replication we monitor the computation time of the individual initial estimators as well as the time needed by the subsequent C-steps procedure. The averaged computation times (in seconds) are given in Table G.3.2.

Table G.3.2: Averaged computation times of the four initial estimators (init) and the subsequent C-steps, in the four data settings described in Section G.3.2.

	Spatial median		SDO		Spatial rank		SSCM	
	init	C-steps	init	C-steps	init	C-steps	init	C-steps
Setting 1	0.020	0.002	0.046	0.002	0.406	0.002	0.034	0.002
Setting 2	0.007	0.001	0.029	0.001	0.070	0.001	0.010	0.001
Setting 3	0.024	0.024	0.055	0.024	0.667	0.025	0.037	0.024
Setting 4	0.023	0.018	0.052	0.028	0.670	0.027	0.035	0.034

In Table G.3.2 we see that the C-steps never take longer than their initial estimator, and that the computation times of the C-steps are similar across the four initial estimates. Among the initial estimators, the spatial rank took the longest, whereas the other three took about the same time. Also note that the bivariate settings 3 and 4 are not faster than setting 1 because their sample size is $n = 500$, so the computations are done on kernel matrices of size 500×500.

We also counted the number of times each initial estimator provided the best solution (i.e. the lowest covariance determinant) after its C-steps converged. The results are given in Table G.3.3. Note that two or more estimators can give the same best solution, so the row sums can exceed 100. In settings 1 and 2 of high-dimensional data analyzed with the linear kernel, the SSCM initial estimator outperformed the others. In settings 3 and 4 with nonlinear kernels, all four initial estimators performed about equally well.

Table G.3.3: The number of times each initial estimator provided the best solution (i.e. the lowest covariance determinant) after C-steps, for the 4 settings in Table G.3.2.

	Spatial median	SDO	Spatial rank	SSCM
Setting 1	4	0	1	95
Setting 2	0	5	0	95
Setting 3	57	65	52	66
Setting 4	52	60	58	53

G.3.3 Additional simulation results with linear kernel

Table G.3.4 shows the mean squared error (MSE) of the estimates $\hat{\Sigma}$ in the same setup as Table 6.2.1 of the main text. For readability the MSE values are multiplied by 1000. Table G.3.5 lists the average values of the regularization parameter ρ in each setting.

Table G.3.4: MSE for Σ of type ALYZ (multiplied by 1000).

	Point contamination Value of h/n			Shift contamination Value of h/n			Cluster contamination Value of h/n		
	0.50	0.75	0.90	0.50	0.75	0.90	0.50	0.75	0.90
$\varepsilon = 0$:									
KMRCD									
400×200	4.31	2.96	2.47	4.33	2.94	2.46	4.35	2.94	2.43
300×200	5.51	3.81	3.19	5.44	3.80	3.18	5.52	3.79	3.21
200×200	7.88	5.47	4.66	7.78	5.54	4.71	7.77	5.42	4.71
200×300	7.36	5.21	4.45	7.29	5.21	4.45	7.37	5.22	4.45
200×400	6.95	5.04	4.32	6.91	4.99	4.31	6.94	5.11	4.29
MRCD									
400×200	4.32	2.99	2.51	4.32	2.96	2.51	4.34	2.99	2.48
300×200	5.59	3.83	3.22	5.51	3.83	3.21	5.54	3.83	3.26
200×200	7.89	5.60	4.67	7.85	5.59	4.69	7.75	5.61	4.70
200×300	7.42	5.28	4.51	7.42	5.28	4.49	7.47	5.33	4.52
200×400	7.04	5.04	4.33	7.08	5.04	4.36	7.01	5.09	4.31
$\varepsilon = 0.1$:									
KMRCD									
400×200	4.35	2.99	2.53	4.38	3.00	2.51	4.37	2.99	2.50
300×200	5.61	3.86	3.29	5.59	3.90	3.28	5.60	3.89	3.27
200×200	7.97	5.60	4.86	7.93	5.58	4.84	7.99	5.64	4.83
200×300	7.38	5.31	4.59	7.44	5.32	4.56	7.39	5.30	4.60
200×400	7.09	5.13	4.39	7.03	5.11	4.41	7.04	5.10	4.43
MRCD									
400×200	4.43	3.05	2.58	4.52	3.12	2.60	4.49	3.11	2.59
300×200	5.75	3.95	3.34	5.85	4.05	3.35	5.88	4.04	3.36
200×200	8.33	5.74	4.94	8.18	5.81	4.97	8.20	5.86	4.96
200×300	7.68	5.45	4.62	7.78	5.57	4.69	7.78	5.52	4.72
200×400	7.22	5.15	4.38	7.44	5.28	4.52	7.41	5.31	4.55
$\varepsilon = 0.3$:									
KMRCD									
400×200	4.55	247.32	58.13	4.55	18.74	40.78	4.54	18.75	39.88
300×200	5.80	242.03	58.59	5.88	18.34	38.22	5.80	18.47	39.34
200×200	8.28	246.56	59.43	8.51	19.15	37.54	8.37	19.09	37.85
200×300	7.78	109.01	26.17	7.88	9.43	16.42	7.87	9.43	16.47
200×400	7.41	61.23	15.02	7.40	6.08	9.44	7.45	6.13	9.49
MRCD									
400×200	4.16	329.39	89.32	4.85	37.14	115.04	4.87	36.75	114.14
300×200	5.25	312.86	90.28	6.39	37.43	109.50	6.31	37.72	114.09
200×200	7.44	326.92	92.79	9.07	38.99	108.50	8.83	39.07	110.41
200×300	6.49	128.66	33.82	8.52	17.44	42.28	8.56	17.53	42.15
200×400	5.62	64.31	16.70	7.97	10.51	21.52	8.03	10.75	21.83

Table G.3.5: Regularization coefficients for Σ of type ALYZ.

	Point contamination Value of h/n			Shift contamination Value of h/n			Cluster contamination Value of h/n		
	0.50	0.75	0.90	0.50	0.75	0.90	0.50	0.75	0.90
$\varepsilon = 0$:									
KMRCD									
400×200	0.091	0.075	0.069	0.093	0.073	0.071	0.088	0.077	0.072
300×200	0.094	0.086	0.079	0.097	0.089	0.079	0.097	0.088	0.087
200×200	0.117	0.096	0.091	0.119	0.100	0.091	0.117	0.098	0.092
200×300	0.155	0.116	0.107	0.155	0.116	0.109	0.162	0.116	0.108
200×400	0.179	0.143	0.125	0.178	0.139	0.124	0.178	0.138	0.126
MRCD									
400×200	0.091	0.073	0.063	0.093	0.070	0.064	0.092	0.073	0.064
300×200	0.100	0.084	0.078	0.098	0.087	0.075	0.099	0.085	0.077
200×200	0.115	0.099	0.094	0.115	0.102	0.094	0.115	0.102	0.092
200×300	0.143	0.116	0.107	0.140	0.119	0.108	0.145	0.117	0.107
200×400	0.164	0.139	0.125	0.169	0.138	0.125	0.165	0.134	0.124
$\varepsilon = 0.1$:									
KMRCD									
400×200	0.089	0.079	0.074	0.089	0.076	0.074	0.095	0.080	0.069
300×200	0.098	0.085	0.079	0.097	0.089	0.083	0.098	0.088	0.079
200×200	0.424	0.098	0.097	0.120	0.099	0.090	0.120	0.098	0.092
200×300	0.156	0.120	0.110	0.153	0.118	0.112	0.150	0.118	0.112
200×400	0.181	0.139	0.127	0.176	0.140	0.126	0.176	0.140	0.125
MRCD									
400×200	0.087	0.073	0.064	0.082	0.061	0.059	0.081	0.065	0.054
300×200	0.096	0.081	0.073	0.092	0.076	0.069	0.089	0.073	0.065
200×200	0.115	0.096	0.091	0.102	0.088	0.078	0.103	0.088	0.081
200×300	0.146	0.118	0.109	0.133	0.105	0.098	0.129	0.107	0.100
200×400	0.172	0.138	0.130	0.153	0.125	0.115	0.154	0.124	0.114
$\varepsilon = 0.3$:									
KMRCD									
400×200	0.094	0.524	0.740	0.097	0.507	0.733	0.091	0.517	0.744
300×200	0.100	0.505	0.738	0.101	0.527	0.746	0.101	0.521	0.742
200×200	0.124	0.507	0.746	0.119	0.511	0.749	0.118	0.521	0.745
200×300	0.151	0.508	0.749	0.153	0.516	0.748	0.155	0.504	0.748
200×400	0.178	0.512	0.740	0.175	0.510	0.737	0.174	0.518	0.748
MRCD									
400×200	0.534	0.644	0.705	0.071	0.308	0.541	0.068	0.314	0.555
300×200	0.459	0.745	0.724	0.078	0.325	0.557	0.080	0.323	0.558
200×200	0.750	0.688	0.712	0.092	0.319	0.565	0.089	0.326	0.564
200×300	0.774	0.770	0.760	0.116	0.349	0.591	0.118	0.338	0.592
200×400	0.796	0.790	0.775	0.139	0.364	0.610	0.142	0.372	0.616

G.4 Additional figures for non-elliptical data

Here we illustrate the Frank, Clayton, and Gumbel copulas [167] with Kendall rank correlation $\tau = 0.6$. Figure G.4.1 below is analogous to Figure 6.2.4 for the t copula in the main text.

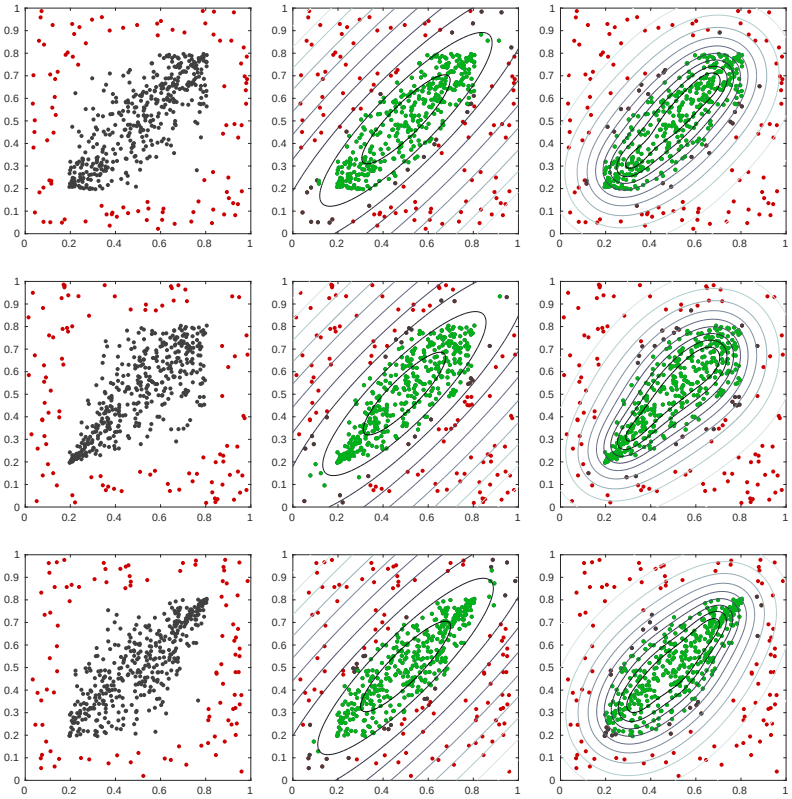


Figure G.4.1: Illustration of the non-elliptical simulation setting, on data sets generated from the Frank (top), Clayton (middle), and Gumbel (bottom) copulas. Each dataset contains 20% of outlying observations. In the left panels, the regular observations are shown in black and the outliers in red. The results of the MRCD estimator are in the middle panels, and those of the KMRCD estimator in the rightmost panels, each for $h = 0.75n$. In those panels the points in the h-subset are shown in green, and the other points with the $n(1-\varepsilon)$ lowest (kernel) Mahalanobis distance are depicted in grey. The remaining points are shown in red. The curves are contours of the robust (kernel) Mahalanobis distance.

Bibliography

[1] AI system for breast cancer screening, Deepmind. https://deepmind.com/research/publications/International-evaluation-of-an-artificial-intelligence-system-to-identify-breast-cancer-in-screening-mammography. Accessed: 10/06/2021.

[2] AlphaGo, Deepmind. https://deepmind.com/research/case-studies/alphago-the-story-so-far. Accessed: 10/06/2021.

[3] AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii. Accessed: 21/07/2021.

[4] What a machine learning tool that turns Obama white can (and can't) tell us about AI bias. https://www.theverge.com/21298762/face-depixelizer-ai-machine-learning-tool-pulse-stylegan-obama-bias. Accessed: 10/06/2021.

[5] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). 2016.

[6] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[7] A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69. PMLR, 2018.

[8] C. Agostinelli, A. Leung, V. J. Yohai, and R. H. Zamar. Robust estimation of multivariate location and scatter in the presence of cellwise and casewise contamination. *Test*, 24:441–461, 2015.

[9] A. E. Alaoui and M. W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, pages 775–783, 2015.

[10] C. Alzate and J. A. K. Suykens. Kernel component analysis using an epsilon-insensitive robust loss function. *IEEE Transactions on Neural Networks*, 19:1583–1598, 2008.

[11] C. Alzate and J. A.K. Suykens. Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):335–347, 2008.

[12] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017.

[13] A. Askari, F. Yang, and L. El Ghaoui. Kernel-based outlier detection using the inverse Christoffel function, arxiv preprint arxiv:1806.06775, 2018.

[14] H. Avron, H. Nguyen, and D. Woodruff. Subspace Embeddings for the Polynomial Kernel. In *Advances in Neural Information Processing Systems*, volume 27, pages 2258–2266, 2014.

[15] F. Bach. Sharp analysis of low-rank kernel matrix approximations. In *COLT Conference on Learning Theory*, pages 185–209, 2013.

[16] R. Bapat and V. Sunder. On majorization and Schur products. *Linear Algebra and its Applications*, 72:107 – 117, 1985.

[17] R. Bardenet and A. Hardy. Monte Carlo with Determinantal Point Processes. *Annals of Applied Probability*, 2019.

[18] M.-A. Belabbas and P. J. Wolfe. Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, 106(2):369–374, 2009.

[19] R. Bhatia, T. Jain, and Y. Lim. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165 – 191, 2019.

[20] A. Bibal and B. Frénay. Interpretability of machine learning models and representations: an introduction. In *ESANN*, 2016.

[21] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, 43(3):1457–1472, 2011.

[22] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD GANs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[23] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[24] K. Boudt, P. J. Rousseeuw, S. Vanduffel, and T. Verdonck. The minimum regularized covariance determinant estimator. *Statistics and Computing*, 30:113–128, 2020.

[25] S. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

[26] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[27] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[28] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

[29] A. T. Bui, J.-K. Im, D. W. Apley, and G. C. Runger. Projection-Free Kernel Principal Component Analysis for Denoising. *Neurocomputing*, 2019.

[30] C. Burgess and H. Kim. 3D Shapes Dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

[31] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in *beta*-VAE. *arXiv preprint arXiv:1804.03599*, 2018.

[32] D. Calandriello, A. Lazaric, and M. Valko. Distributed adaptive sampling for kernel matrix approximation. In A. Singh and X. J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1421–1429. PMLR, 2017.

[33] D. Calandriello, A. Lazaric, and M. Valko. Second-order kernel online convex optimization with adaptive sketching. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 645–653. JMLR. org, 2017.

[34] A. Camuto, M. Willetts, S. Roberts, C. Holmes, and T. Rainforth. Towards a theoretical understanding of the robustness of variational autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 3565–3573. PMLR, 2021.

[35] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[36] C. Cath. Governing artificial intelligence: ethical, legal and technical opportunities and challenges., 2018.

[37] S. Caton and C. Haas. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053*, 2020.

[38] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode Regularized Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[39] L. Chen, H. Wang, J. Zhao, P. Koutris, and D. Papailiopoulos. The effect of network width on the performance of large-batch training. *Advances in neural information processing systems*, 2018.

[40] L. Chen, G. Zhang, and H. Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5627–5638, 2018.

[41] M. Chen and L. Denoyer. Multi-view generative adversarial networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 175–188. Springer, 2017.

[42] R. T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating Sources of Disentanglement in Variational Autoencoders. In *Advances in Neural Information Processing Systems 31*, pages 2610–2620, 2018.

[43] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[44] Y. Chen, X. Dang, H. Peng, and H. L. Bart. Outlier detection with the kernelized spatial depth function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):288–305, 2008.

[45] G. G. Chrysos, J. Kossaifi, and S. Zafeiriou. Robust conditional generative adversarial networks. *International Conference on Learning Representations*, 2019.

[46] A. Civril and M. Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47):4801 – 4811, 2009.

[47] M. B. Cohen, C. Musco, and J. Pachocki. Online row sampling. In *Proceedings of the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2015.

[48] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. In *Advances in computational mathematics*, pages 329–359, 1996.

[49] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[50] C. Croux and G. Haesbroeck. Influence function and efficiency of the minimum covariance determinant scatter matrix estimator. *Journal of Multivariate Analysis*, 71:161–190, 1999.

[51] S. Curi, K. Levy, S. Jegelka, A. Krause, et al. Adaptive sampling for stochastic risk-averse learning. *Advances in Neural Information Processing Systems*, 2020.

[52] S. Dasgupta and A. Gupta. An Elementary Proof of a Theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, Jan. 2003.

[53] B. De Ketelaere, M. Hubert, J. Raymaekers, P. J. Rousseeuw, and I. Vranckx. Real-time outlier detection for large datasets by RT-DetMCD. *Chemometrics and Intelligent Laboratory Systems*, 199:103957, 2020.

[54] M. Debruyne, M. Hubert, and J. Van Horebeek. Detecting influential observations in kernel PCA. *Computational Statistics & Data Analysis*, 54:3007–3019, 2010.

[55] M. Debruyne, S. Serneels, and T. Verdonck. Robustified least squares support vector classification. *Journal of Chemometrics*, 23:479–486, 2009.

[56] M. Debruyne and T. Verdonck. Robust kernel principal component analysis and classification. *Advances in Data Analysis and Classification*, 4:151–167, 2010.

[57] M. Dereziński, R. Khanna, and M. W. Mahoney. Improved guarantees and a multiple-descent curve for column subset selection and the nystrom method. *Advances in Neural Information Processing Systems*, 33, 2020.

[58] M. Dereziński, F. T. Liang, and M. W. Mahoney. Exact expressions for double descent and implicit regularization via surrogate random design. *Advances in Neural Information Processing Systems*, 33, 2020.

[59] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1117–1126. Society for Industrial and Applied Mathematics, 2006.

[60] A. Desolneux, C. Launay, and B. Galerne. Exact Sampling of Determinantal Point Processes without Eigendecomposition, hal-01710266v4. 2017.

[61] R. DeVore, G. Petrova, and P. Wojtaszczyk. Greedy algorithms for reduced bases in Banach spaces. *Constructive Approximation*, 37(3):455–466, Jun 2013.

[62] A. B. Dieng, F. J. R. Ruiz, D. M. Blei, and M. K. Titsias. Prescribed generative adversarial networks. *arxiv:1910.04302*, 2020.

[63] M. Diesendruck, E. R. Elenberg, R. Sen, G. W. Cole, S. Shakkottai, and S. A. Williamson. Importance weighted generative networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 249–265, 2019.

[64] A. N. Dolia, T. De Bie, C. J. Harris, J. Shawe-Taylor, and D. M. Titterington. The minimum volume covering ellipsoid estimation in kernel-defined feature spaces. In *European Conference on Machine Learning*, pages 630–637. Springer, 2006.

[65] A. N. Dolia, C. J. Harris, J. Shawe-Taylor, and D. M. Titterington. Kernel ellipsoidal trimming. *Computational Statistics & Data Analysis*, 52:309–324, 2007.

[66] D. L. Donoho. Breakdown properties of multivariate location estimators. Technical report, Harvard University, Boston, 1982.

[67] D. Dowson and B. Landau. The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450 – 455, 1982.

[68] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012.

[69] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[70] E. Dupont. Learning disentangled joint continuous and discrete representations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 708–718, 2018.

[71] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[72] C. Eastwood and C. K. I. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[73] A. El Alaoui and M. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems 28*, pages 775–783, 2015.

[74] M. Elfeki, C. Couprie, M. Riviere, and M. Elhoseiny. GDPP: Learning Diverse Generations using Determinantal Point Processes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[75] V. Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[76] A. Fischer and C. Igel. Training restricted Boltzmann machines: An introduction. *Pattern Recognition*, 47(1):25–39, 2014.

[77] L. Floridi, J. Cowls, M. Beltrametti, R. Chatila, P. Chazerand, V. Dignum, C. Luetge, R. Madelin, U. Pagallo, F. Rossi, et al. AI4People—an ethical framework for a good AI society: opportunities, risks, principles, and recommendations. *Minds and Machines*, 28(4):689–707, 2018.

[78] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[79] F. Futami, I. Sato, and M. Sugiyama. Variational inference based on robust divergences. *21st International Conference on Artificial Intelligence and Statistics*, 2018.

[80] T. Gao, S. Kovalsky, and I. Daubechies. Gaussian process landmarking on manifolds. *SIAM Journal on Mathematics of Data Science*, 1(1):208–236, 2019.

[81] M. Gelbrich. On a Formula for the L2 Wasserstein Metric between Measures on Euclidean and Hilbert spaces. *Mathematische Nachrichten*, 147(1):185–203, 1990.

[82] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[83] A. Ghosh, V. Kulharia, V. P. Namboodiri, P. H. Torr, and P. K. Dokania. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[84] P. Ghosh, M. S. Sajjadi, A. Vergari, M. Black, and B. Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

[85] J. Gillenwater, A. Kulesza, and B. Taskar. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 2735–2743, 2012.

[86] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[87] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[88] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in NIPS*, pages 2672–2680, 2014.

[89] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[90] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13:723–773, 2012.

[91] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. In *Advances in neural information processing systems 31*, 2017.

[92] V. Guruswami and A. K. Sinop. Optimal column-based low-rank matrix reconstruction. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, page 1207–1214, 2012.

[93] B. Haasdonk and E. Pekalska. Classification with kernel mahalanobis distance classifiers. In *Advances in Data Analysis, Data Handling and Business Intelligence*, pages 351–361. Springer, 2009.

[94] M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29:3315–3323, 2016.

[95] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer New York Inc., New York, NY, USA, 2001.

[96] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[97] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[98] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems 30*, 2017.

[99] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations*, 2017.

[100] G. Hinton. A practical guide to training restricted Boltzmann machines. *Momentum*, 9(1):926, 2010.

[101] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[102] H. Hoffmann. Kernel PCA for novelty detection. *Pattern recognition*, 40(3):863–874, 2007.

[103] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[104] P. Honeine and C. Richard. Preimage Problem in Kernel-Based Machine Learning. *IEEE Signal Processing Magazine*, 28(2):77–88, Mar. 2011.

[105] Y. Hong and R. A. Horn. The Jordan Canonical Form of a Product of a Hermitian and a Positive Semidefinite Matrix. *Linear Algebra and its Applications*, 147:373 – 386, 1991.

[106] J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág. Determinantal processes and independence. *Probab. Surveys*, 3:206–229, 2006.

[107] C.-J. Hsieh, S. Si, and I. Dhillon. A divide-and-conquer solver for kernel support vector machines. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 566–574, 2014.

[108] M. Hubert and M. Debruyne. Minimum covariance determinant. *Wiley interdisciplinary reviews: Computational statistics*, 2(1):36–43, 2010.

[109] M. Hubert, M. Debruyne, and P. J. Rousseeuw. Minimum Covariance Determinant and extensions. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(3):e1421, 2018.

[110] M. Hubert, P. Rousseeuw, and S. Van Aelst. High breakdown robust multivariate methods. *Statistical Science*, 23:92–119, 2008.

[111] M. Hubert, P. J. Rousseeuw, and K. Vanden Branden. ROBPCA: a new approach to robust principal component analysis. *Technometrics*, 47:64–79, 2005.

[112] M. Hubert, P. J. Rousseeuw, and T. Verdonck. A deterministic algorithm for robust location and scatter. *Journal of Computational and Graphical Statistics*, 21:618–637, 2012.

[113] M. Hubert and K. Van Driessen. Fast and robust discriminant analysis. *Computational Statistics & Data Analysis*, 45:301–320, 2004.

[114] Z. Ji, Z. C. Lipton, and C. Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.

[115] C. Johnson and H. Robinson. Eigenvalue inequalities for principal submatrices. *Linear Algebra and its Applications*, 37:11 – 22, 1981.

[116] I. T. Jolliffe. *Principal component analysis*. Springer, 1986.

[117] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[118] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[119] D. S. Kershaw. The incomplete Cholesky—conjugate gradient method for the iterative solution of systems of linear equations. *Journal of computational physics*, 26(1):43–65, 1978.

[120] H. Kim and A. Mnih. Disentangling by Factorising. In *proceedings of the Thirty-fifth International Conference on Machine Learning (ICML)*, volume 80, pages 2649–2658, 2018.

[121] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[122] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[123] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[124] A. Kulesza and B. Taskar. k-DPPs: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

[125] A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

[126] S. Kumar, M. Mohri, and A. Talwalkar. Ensemble nystrom method. In *Advances in Neural Information Processing Systems*, pages 1060–1068, 2009.

[127] J. T. Kwok and I. W.-H. Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15:1517–1525, 2003.

[128] H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. In *Proceedings of the 25th International Conference on Machine Learning - ICML '08*, pages 536–543, Helsinki, Finland, 2008. ACM Press.

[129] J. Lasserre and E. Pauwels. The empirical Christoffel function with applications in Machine Learning, Arxiv preprint arXiv:1701.02886, 2017.

[130] C. Launay, B. Galerne, and A. Desolneux. Exact sampling of determinantal point processes without eigendecomposition. *Journal of Applied Probability*, 57(4):1198–1221, 2020.

[131] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[132] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[133] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–97–104 Vol.2, 2004.

[134] K. S. Lee, N.-T. Tran, and N.-M. Cheung. InfoMax-GAN: Mutual information maximization for improved adversarial image generation. In *NeurIPS 2019 Workshop on Information Theory and Machine Learning*, 2019.

[135] C. Li, S. Jegelka, and S. Sra. Efficient sampling for k-determinantal point processes. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1328–1337, 2016.

[136] C. Li, S. Jegelka, and S. Sra. Fast DPP sampling for Nyström with application to kernel methods. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pages 2061–2070, 2016.

[137] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Poczos. MMD GAN: Towards Deeper Understanding of Moment Matching Network. In *Advances in Neural Information Processing Systems 30*, 2017.

[138] J. Li, F. Li, and S. Todorovic. Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform. In *proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[139] D. Liang and J. Paisley. Landmarking manifolds with Gaussian processes. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 466–474, 2015.

[140] S.-B. Lin, X. Guo, and D.-X. Zhou. Distributed learning with regularized least squares. *The Journal of Machine Learning Research*, 18(1):3202–3232, 2017.

[141] Z. Lin, A. Khetan, G. Fanti, and S. Oh. Pacgan: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 1498–1507, 2018.

[142] Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[143] M.-Y. Liu and O. Tuzel. Coupled Generative Adversarial Networks. In *Advances in Neural Information Processing Systems 29*, pages 469–477. Curran Associates, Inc., 2016.

[144] Y. Liu, E. Jun, Q. Li, and J. Heer. Latent space cartography: Visual analysis of vector space embeddings. In *Computer Graphics Forum*, volume 38, pages 67–78. Wiley Online Library, 2019.

[145] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[146] F. Locatello, S. Bauer, M. Lučić, G. Rätsch, S. Gelly, B. Schölkopf, and O. F. Bachem. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *International Conference on Machine Learning (ICML)*, 2019.

[147] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem. Disentangling Factors of Variations Using Few Labels. In *International Conference on Learning Representations (ICLR)*, 2020.

[148] D. G. Lowe et al. Object recognition from local scale-invariant features. In *ICCV*, volume 99, pages 1150–1157, 1999.

[149] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? a large-scale study. In *Advances in neural information processing systems*, pages 700–709, 2018.

[150] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

[151] P. Ma, M. Mahoney, and B. Yu. A statistical perspective on algorithmic leveraging. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 91–99, 2014.

[152] R. A. Maronna and R. H. Zamar. Robust estimates of location and dispersion for high-dimensional datasets. *Technometrics*, 44:307–317, 2002.

[153] E. Massart and P.-A. Absil. Quotient Geometry with Simple Geodesics for the Manifold of Fixed-Rank Positive-Semidefinite Matrices. *SIAM Journal on Matrix Analysis and Applications*, 41(1):171–198, 2020.

[154] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. dSprites: Disentanglement testing Sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[155] S. McCurdy. Ridge regression and provable deterministic ridge leverage score sampling. In *Advances in Neural Information Processing Systems 31*, pages 2468–2477, 2018.

[156] L. McInnes, J. Healy, N. Saul, and L. Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[157] J. Mercer. Functions of Positive and Negative Type, and Their Connection the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209(441-458):415–446, Jan. 1909.

[158] L. Mescheder, S. Nowozin, and A. Geiger. The numerics of GANs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 1823–1833, 2017.

[159] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. *ICLR*, 2017.

[160] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and denoising in feature spaces. In *Advances in neural information processing systems*, pages 536–542, 1999.

[161] S. Minsker. On some extensions of Bernstein's inequality for self-adjoint operators. *Statistics & Probability Letters*, 127:111 – 119, 2017.

[162] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[163] G. B. Moody and R. G. Mark. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[164] Y. Mroueh, T. Sercu, and V. Goel. McGan: Mean and Covariance Feature Matching GAN. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[165] C. Musco and C. Musco. Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems*, pages 3833–3845, 2017.

[166] M. Mutný, M. Dereziński, and A. Krause. Convergence analysis of block coordinate algorithms with determinantal sampling. In *The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[167] R. B. Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007.

[168] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 1st edition, 2014.

[169] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[170] T. Nguyen, T. Le, H. Vu, and D. Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[171] J. H. Oh, M. Pouryahya, A. Iyer, A. P. Apte, J. O. Deasy, and A. Tannenbaum. A novel kernel Wasserstein distance on Gaussian measures: An application of identifying dental artifacts in head and neck computed tomography. *Computers in Biology and Medicine*, 120:103731, 2020.

[172] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. Dota 2 with large scale deep reinforcement learning. 2019.

[173] S. Oymak, B. Recht, and M. Soltanolkotabi. Isometric sketching of any set via the Restricted Isometry Property. *Information and Inference: A Journal of the IMA*, 7(4):707–726, 03 2018.

[174] J. Paisley, X. Liao, and L. Carin. Active learning and basis selection for kernel-based linear models: A Bayesian perspective. *IEEE Transactions on Signal Processing*, 58(5):2686–2700, 2010.

[175] A. Pandey, J. Schreurs, and J. A. Suykens. Generative restricted kernel machines: A framework for multi-view generation and disentangled feature learning. *Neural Networks*, 135:177–191, 2021.

[176] D. Papailiopoulos, A. Kyrillidis, and C. Boutsidis. Provable deterministic leverage score sampling. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 997–1006, 2014.

[177] E. Pauwels, F. Bach, and J.-P. Vert. Relating Leverage Scores and Density using Regularized Christoffel Functions. In *Neural Information Processing Systems*, 2018.

[178] R. Pemantle and Y. Peres. Concentration of Lipschitz functionals of determinantal and other strong Rayleigh measures. *Combinatorics, Probability and Computing*, 23(1):140–160, 2014.

[179] G. Peyré, M. Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[180] R. Peyre. Comparison between W_2 distance and $\dot{H}^{-1}$ norm, and Localization of Wasserstein distance. *ESAIM: COCV*, 24(4):1489–1501, 2018.

[181] A. Pinkus. Matrices and n-widths. *Linear Algebra and Applications*, 27:245–278, 1979.

[182] J. Poulson. High-performance sampling of generic determinantal point processes. *Philosophical Transactions of the Royal Society A*, 378(2166):20190059, 2020.

[183] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational Autoencoder for Deep Learning of Images, Labels and Captions. NIPS'16, pages 2360–2368, USA, 2016. Curran Associates Inc.

[184] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[185] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1177–1184, 2007.

[186] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, 2006.

[187] S. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep Visual Analogy-Making. In *Advances in Neural Information Processing Systems*, 2015.

[188] Y. Ren, J. Zhu, J. Li, and Y. Luo. Conditional Generative Moment-Matching Networks. In *Advances in Neural Information Processing Systems 29*, 2016.

[189] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[190] R. T. Rockafellar. *Conjugate Duality and Optimization*. SIAM, 1974.

[191] M. Rolinek, D. Zietlow, and G. Martius. Variational autoencoders pursue pca directions (by accident). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

[192] P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880, 1984.

[193] P. J. Rousseeuw. Multivariate estimation with high breakdown point. In W. Grossmann, G. Pflug, I. Vincze, and W. Wertz, editors, *Mathematical Statistics and Applications*, pages 283–297. Reidel, 1985.

[194] P. J. Rousseeuw and C. Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88:1273–1283, 1993.

[195] P. J. Rousseeuw and M. Hubert. Anomaly detection by robust statistics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2):e1236, 2018.

[196] P. J. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. Wiley-Interscience, New York, 1987.

[197] P. J. Rousseeuw, J. Raymaekers, and M. Hubert. A measure of directional outlyingness with applications to image data and video. *Journal of Computational and Graphical Statistics*, 27:345–359, 2018.

[198] P. J. Rousseeuw, S. Van Aelst, K. Van Driessen, and J. A. Gulló. Robust multivariate regression. *Technometrics*, 46(3):293–305, 2004.

[199] P. J. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 1999.

[200] A. Rudi, D. Calandriello, L. Carratino, and L. Rosasco. On fast leverage score sampling and optimal learning. In *Advances in Neural Information Processing Systems*, pages 5673–5683, 2018.

[201] A. Rudi, L. Carratino, and L. Rosasco. Falkon: An optimal large scale kernel method. In *Advances in Neural Information Processing Systems 30*, pages 3888–3898, 2017.

[202] A. Rudi, E. De Vito, A. Verri, and F. Odone. Regularized kernel algorithms for support estimation. *Frontiers in Applied Mathematics and Statistics*, 3:23, 2017.

[203] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 5234–5243, 2018.

[204] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. In *Advances in neural information processing systems 29*, 2016.

[205] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[206] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[207] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[208] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[209] B. Schölkopf, A. J. Smola, F. Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[210] J. Schreurs and J. A. K. Suykens. Generative Kernel PCA. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* , pages 129–134, 2018.

[211] J. Schreurs, I. Vranckx, M. Hubert, J. A. Suykens, and P. J. Rousseeuw. Outlier detection in non-elliptical data by kernel MRCD. *Statistics and Computing*, 31(5):1–18, 2021.

[212] H. Sedghamiz. An online algorithm for r, s and t wave detection. *Linkoping University, Dec*, 2013.

[213] B. Shneiderman. Bridging the gap between ethics and practice: Guidelines for reliable, safe, and trustworthy human-centered ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 10(4):1–31, 2020.

[214] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[215] S. Sinha, H. Zhang, A. Goyal, Y. Bengio, H. Larochelle, and A. Odena. Small-GAN: Speeding up GAN training using core-sets. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 9005–9015, 2020.

[216] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11918–11930, 2019.

[217] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. Veegan: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems 30*, 2017.

[218] N. Srivastava and R. R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[219] W. Stahel. Breakdown of covariance estimators. Technical report, Fachgruppe für Statistik, ETH Zürich, 1981.

[220] N. Sterge, B. Sriperumbudur, L. Rosasco, and A. Rudi. Gain with no pain: Efficiency of kernel-PCA by Nyström sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 3642–3652. PMLR, 2020.

[221] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[222] D. Stutz, M. Hein, and B. Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

[223] J. A. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor. A support vector machine formulation to PCA analysis and its kernel version. *IEEE Transactions on Neural Networks*, 14:447–450, 2003.

[224] J. A.K. Suykens. Deep Restricted Kernel Machines using Conjugate Feature Duality. *Neural Computation*, 29(8):2123–2163, Aug. 2017.

[225] J. A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. River Edge, NJ, Jan. 2002.

[226] M. Suzuki, K. Nakayama, and Y. Matsuo. Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*, 2016.

[227] Y. Tang, R. Salakhutdinov, and G. Hinton. Robust Boltzmann machines for recognition and denoising. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2264–2271. IEEE, 2012.

[228] I. O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf. Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems*, pages 5424–5433, 2017.

[229] E. Toreini, M. Aitken, K. Coopamootoo, K. Elliott, C. G. Zelaya, and A. Van Moorsel. The relationship between trust in AI and trustworthy machine learning technologies. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 272–283, 2020.

[230] L. Tran, X. Yin, and X. Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1415–1424, 2017.

[231] N. Tremblay, S. Barthelme, and P.-O. Amblard. Optimized algorithms to sample determinantal point processes. *arXiv preprint arXiv:1802.08471*, 2018.

[232] N. Tremblay, S. Barthelmé, and P.-O. Amblard. Determinantal point processes for coresets. *Journal of Machine Learning Research*, 20(168):1–70, 2019.

[233] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33, 2020.

[234] J. Tropp. Freedman's inequality for matrix martingales. *Electron. Commun. Probab.*, 16:262–270, 2011.

[235] A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

[236] T. Van Steenkiste, D. Deschrijver, and T. Dhaene. Interpretable ECG beat embedding using disentangled variational auto-encoders. In *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 373–378. IEEE, 2019.

[237] Y. Vardi and C.-H. Zhang. The multivariate $l1$-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97(4):1423–1426, 2000.

[238] S. Visuri, V. Koivunen, and H. Oja. Sign and rank covariance matrices. *Journal of Statistical Planning and Inference*, 91:557–575, 2000.

[239] A. Voynov and A. Babenko. Unsupervised discovery of interpretable directions in the gan latent space. In *International Conference on Machine Learning*, pages 9786–9796. PMLR, 2020.

[240] S. Wang, A. Gittens, and M. W. Mahoney. Scalable kernel K-means clustering with Nyström approximation: relative-error bounds. *The Journal of Machine Learning Research*, 20(1):431–479, 2019.

[241] W. Wang, Y. Sun, and S. Halgamuge. Improving MMD-GAN training with repulsive loss function. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[242] G. P. Way and C. S. Greene. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *BioRxiv*, page 174474, 2017.

[243] J. Weston, B. Schölkopf, and G. H. Bakir. Learning to Find Pre-Images. In *NIPS 16*, pages 449–456. 2004.

[244] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688, 2001.

[245] D. Winant, J. Schreurs, and J. A. Suykens. Latent space exploration using generative kernel PCA. In *Bogaerts B. et al. (eds) Artificial Intelligence and Machine Learning. Communications in Computer and Information Science, vol 1196. Springer, Cham. (BNAIC 2019, Benelearn 2019)*, pages 70–82. Springer, 2019.

[246] B. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro. Learning non-discriminatory predictors. In *Conference on Learning Theory*, pages 1920–1953. PMLR, 2017.

[247] J. Wu, Z. Huang, J. Thoma, D. Acharya, and L. Van Gool. Wasserstein Divergence for GANs. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018.

[248] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in NeurIPS*, pages 82–90, 2016.

[249] M. Wu and N. Goodman. Multimodal generative models for scalable weakly-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5575–5585, 2018.

[250] C. Xiao, P. Zhong, and C. Zheng. BourGAN: Generative Networks with Metric Embeddings. In *Advances in Neural Information Processing Systems 32*, 2018.

[251] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. 2017.

[252] Y. Yang, M. Pilanci, and M. J. Wainwright. Randomized sketches for kernels: Fast and optimal nonparametric regression. *The Annals of Statistics*, 45(3):991–1023, 2017.

[253] Z. Yang, M. Moczulski, M. Denil, N. De Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

[254] Y. Yuan and K. M. Kitani. Diverse trajectory forecasting with determinantal point processes. In *International Conference on Learning Representations*, 2019.

[255] M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970. PMLR, 2017.

[256] C. Zhang, H. Kjellstrom, and S. Mandt. Determinantal point processes for mini-batch diversification. *Uncertainty in Artificial Intelligence*, 2017.

[257] C. Zhang, C. Öztireli, S. Mandt, and G. Salvi. Active mini-batch sampling using repulsive point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5741–5748, 2019.

[258] H. Zhang, Z. Zhang, A. Odena, and H. Lee. Consistency regularization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[259] Q.-s. Zhang and S.-c. Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

[260] T. Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.

[261] Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression. In *Conference on learning theory (COLT)*, pages 592–617, 2013.

[262] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable augmentation for data-efficient gan training. *Neural Information Processing Systems*, 2020.

[263] P. Zhong, Y. Mo, C. Xiao, P. Chen, and C. Zheng. Rethinking generative mode coverage: A pointwise guaranteed approach. In *Advances in Neural Information Processing Systems*, pages 2088–2099, 2019.

Curriculum

Joachim Schreurs was born on October 13, 1994 in Hasselt, Belgium.

Degrees & education

PhD student in machine learning, KU Leuven 2017 – 2021
PhD student at the department of electrical engineering under the supervision of Prof. Johan Suykens.

MSc in mathematical engineering, KU Leuven 2015 – 2017
Master thesis: "Updating and tracking of tensor decompositions".

BSc in electrical engineering & computer science, KU Leuven 2012 – 2015
Bachelor project: "Design of a remote-controlled self-balancing robot".

Mathematics & Natural Sciences, Kindsheid Jesu, Hasselt 2010 – 2012

Latin & Mathematics, Kindsheid Jesu, Hasselt 2006 – 2010

List of publications

Articles in peer-reviewed journals

Published

- Joachim Schreurs, Iwein Vranckx, Bart De Ketelaere, Mia Hubert, Johan AK Suykens, and Peter J. Rousseeuw. Outlier detection in non-elliptical data by kernel MRCD. In *Statistics and Computing*, vol. 31, no. 5, 2021.
- Michaël Fanuel, Joachim Schreurs, and Johan AK Suykens. Diversity sampling is an implicit regularization for kernel methods. In *SIAM Journal on Mathematics of Data Science (SIMODS), vol. 3, no. 1, pp. 280-297*, 2021.
- Arun Pandey, Joachim Schreurs, and Johan AK Suykens. Generative restricted kernel machines: A framework for multi-view generation and disentangled feature learning. In *Neural Networks*, 135:177–191, 2021

Under review

- Michaël Fanuel, Joachim Schreurs, and Johan AK Suykens. Determinantal Point Processes Implicitly Regularize Semi-parametric Regression Problems. In *arXiv preprint arXiv:2011.06964*, 2020
- Arun Pandey, Michael Fanuel, Joachim Schreurs, and Johan AK Suykens. Disentangled representation learning and generation with manifold optimization. In *arXiv preprint arXiv:2006.07046*, 2020
- Michaël Fanuel, Joachim Schreurs, and Johan AK Suykens. Nyström landmark sampling and regularized Christoffel functions. In *arXiv preprint arXiv:1905.12346*, 2019.

Published articles in peer-reviewed conference proceedings

- Joachim Schreurs, Hannes De Meulenmeester, Michaël Fanuel, Bart De Moor, and Johan AK Suykens. Leverage Score Sampling for Complete Mode Coverage in Generative Adversarial Networks. In *International Conference on Machine Learning, Optimization, and Data Science (LOD)*, 2021.

- Hannes De Meulenmeester, Joachim Schreurs, Michaël Fanuel, Bart De Moor, and Johan AK Suykens. The Bures Metric for Generative Adversarial Networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2021.

- Joachim Schreurs, Michaël Fanuel, and Johan AK Suykens. Ensemble Kernel Methods, Implicit Regularization and Determinantal Point Processes. In *ICML 2020 workshop on Negative Dependence and Submodularity for ML*, 2020.

- Arun Pandey, Joachim Schreurs, and Johan AK Suykens. Robust generative restricted kernel machines using weighted conjugate feature duality. In *International Conference on Machine Learning, Optimization, and Data Science (LOD)*, pages 613–624. Springer, 2020

- Joachim Schreurs, Michaël Fanuel, and Johan AK Suykens. Towards deterministic diverse subset sampling. In *Proc. of Bogaerts B. et al. (eds) Artificial Intelligence and Machine Learning. Communications in Computer and Information Science, vol 1196. Springer, Cham. (BNAIC 2019, Benelearn 2019)*, pages 137-151, 2019

- David Winant, Joachim Schreurs, and Johan AK Suykens. Latent space exploration using generative kernel PCA. In *Proc. of Bogaerts B. et al. (eds) Artificial Intelligence and Machine Learning. Communications in Computer and Information Science, vol 1196. Springer, Cham. (BNAIC 2019, Benelearn 2019)*, pages 70–82. Springer, 2019

- Joachim Schreurs and Johan A. K. Suykens. Generative Kernel PCA. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 129–134, 2018



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT-STADIUS
Celestijnenlaan 200A box 2402
B-3001 Leuven
https://www.esat.kuleuven.be/stadius/