arXiv:2002.01180v1 [stat.ML] 4 Feb 2020

Robust Generative Restricted Kernel Machines using Weighted Conjugate Feature Duality

Arun Pandey*, Joachim Schreurs*, Johan A. K. Suykens *IEEE Fellow*
Department of Electrical Engineering, ESAT-STADIUS,
KU Leuven. Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
{arun.pandey,joachim.schreurs,johan.suykens}@esat.kuleuven.be

February 5, 2020

Abstract

In the past decade, interest in generative models has grown tremendously. However, their training performance can be highly affected by contamination, where outliers are encoded in the representation of the model. This results in the generation of noisy data. In this paper, we introduce a weighted conjugate feature duality in the framework of Restricted Kernel Machines (RKMs). This formulation is used to fine-tune the latent space of generative RKMs using a weighting function based on the Minimum Covariance Determinant, which is a highly robust estimator of multivariate location and scatter. Experiments show that the weighted RKM is capable of generating clean images when contamination is present in the training data. We further show that the robust method also preserves uncorrelated feature learning through qualitative and quantitative experiments on standard datasets.

***Keywords*—** Machine Learning, Generative Models, Robustness, Kernel Methods, Restricted Kernel Machines

1 Introduction

Generative modeling is an important direction within machine learning, finding applications in image generation [1], anomaly detection [2], denoising [3], collaborative filtering [4] and many more. A popular choice for generation are latent variable models like Variational Auto-Encoders (VAEs) [5] and Restricted Boltzmann Machines (RBMs) [6, 7]. These latent spaces provide a representation of data by embedding the data into an underlying vector space. Exploring these spaces allows for deeper insights in the structure of the data distribution, as well as understanding relationships between data points. The interpretability of the latent space is enhanced when the model learns a disentangled representation [8, 9]. In a disentangled representation, a single latent variable is sensitive to changes in a single generative factor, while being relatively invariant to changes in other factors [10]. For example hair color, lighting conditions or orientation of faces. Another example of generative models are based on adversarial training such as Generative Adversarial Networks (GANs) [1, 11, 12].

In machine learning, training data is often assumed to be ground truth, therefore outliers can severely degrade learned representations and performance of trained models. The same issue arises in generative modelling where contamination of the training data results in 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 [13, 3, 14, 15]. However, these generative models require clean training data or only consider the case where there is label noise. In this paper, we tackle 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 behaviour. We apply methods from traditional robust statistics to the Restricted Kernel Machine (RKM) framework [16] to achieve a robust generative RKM model. The RKM framework yields a representation of kernel methods with visible and hidden units establishing links between kernel methods [17] and RBMs. [18] showed how kernel PCA fits into the RKM framework. A tensor-based multi-view classification model was developed in [19]. In [20], a multi-view generative model called Generative RKM (Gen-RKM) is introduced which uses explicit feature-maps in a novel training procedure for joint feature-selection and subspace learning. Gen-RKM learns an orthogonal latent space which makes it possible to generate data with specific characteristics, i.e. a disentangled representation.

Contributions. This paper introduces a weighted Gen-RKM mechanism that penalizes the outliers and regularizes the latent space. Due to the introduction of a weighted conjugate feature duality, a RKM formulation for weighted kernel PCA is proposed.

*Authors contributed equally.

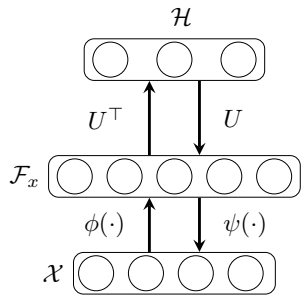


Figure 1: Schematic representation of kernel PCA in the RKM setting. The feature map and pre-image map correspond to ϕ and ψ respectively. The interconnection matrix $\boldsymbol{U}$ models dependencies between latent variables and the input data.

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) [21] is proposed. Qualitative and quantitative experiments on standard datasets show that the proposed model is unaffected by large contamination.

This paper is organized as follows. In Section 2, we briefly discuss the RKM framework and introduce the weighted conjugate feature duality. In Section 3, the weighting scheme is introduced. In Section 4, we show experimental results of our model applied on various public datasets. Section 5 concludes the paper.

2 Weighted Restricted Kernel Machines

2.1 Kernel PCA and Conjugate Feature Duality

We start by introducing kernel PCA in the RKM setting [22, 16]. Take a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, with $\boldsymbol{x}_i \in \mathbb{R}^d$ consisting of N data points. Applying the feature-map $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d_f}$ to the input data point, the Least-Squares Support Vector Machine (LS-SVM) formulation of kernel PCA [17] is written as:

$$\begin{aligned} &\min_{\boldsymbol{U},\boldsymbol{e}_i} \frac{\eta}{2}\operatorname{Tr}(\boldsymbol{U}^\top\boldsymbol{U}) - \frac{1}{2\lambda}\sum_{i=1}^N \boldsymbol{e}_i^\top\boldsymbol{e}_i \\ &\text{s.t. } \boldsymbol{e}_i = \boldsymbol{U}^\top\phi(\boldsymbol{x}_i), \quad \forall i = 1,\ldots,N, \end{aligned} \tag{1}$$

where $\boldsymbol{U} \in \mathbb{R}^{d_f \times s}$ is the unknown interconnection matrix. By the use of *conjugate feature duality*, introduced in [16], the error variables $\boldsymbol{e}_i$ are conjugated to latent variables $\boldsymbol{h}_i$ using:

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

This is also known as the Fenchel-Young inequality in the case of quadratic functions [23]. The variables $\boldsymbol{e}_i$ are eliminated from equation 1 and using equation 2, the RKM training objective function is equal to*:

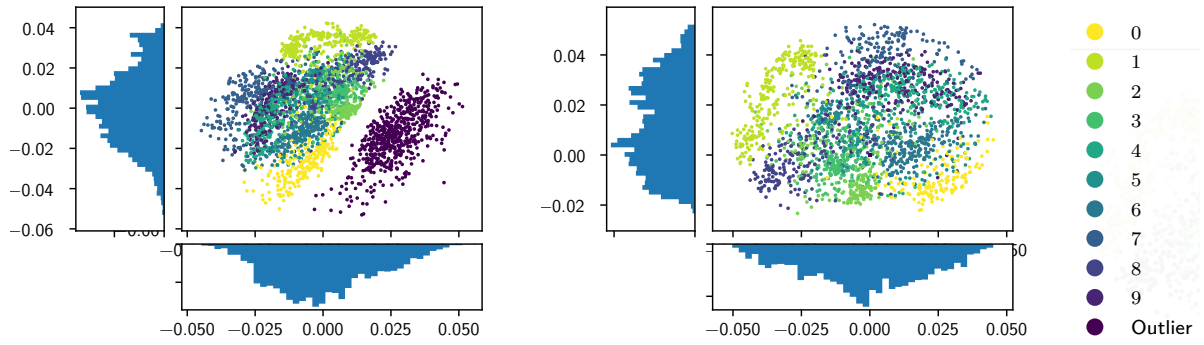
$$\mathcal{J}_t = \sum_{i=1}^N \left(-\phi(\boldsymbol{x}_i)^\top\boldsymbol{U}\boldsymbol{h}_i + \frac{\lambda}{2}\boldsymbol{h}_i^\top\boldsymbol{h}_i\right) + \frac{\eta}{2}\operatorname{Tr}(\boldsymbol{U}^\top\boldsymbol{U}), \tag{3}$$

where $\boldsymbol{h}_i \in \mathbb{R}^s$ are the latent variables modeling the subspace $\mathcal{H}$. Note that the objective function $\mathcal{J}_t$ has an energy form similar to RBMs [24] with additional regularization terms. In the RKM setting, the latent space dimension has a similar interpretation as the number of hidden units in a restricted Boltzmann machine, where in the case of the RKM the hidden units are uncorrelated. Similar to Energy-Based Models [24], the RKM objective function captures dependencies between visible units (training data) and hidden units (latent space) by associating a scalar energy to each configuration of the variables. Learning consists of finding an energy function in which the observed configurations have the lowest energy.

Given $\eta > 0$ as regularization parameters and denoting $\boldsymbol{\Lambda} = \operatorname{diag}\{\lambda_1,\ldots,\lambda_s\} \in \mathbb{R}^{s\times s}$ with $s \leq N$, the solution of above equation yields the following eigenvalue problem:

$$\frac{1}{\eta}\boldsymbol{K}\boldsymbol{H}^\top = \boldsymbol{H}^\top\boldsymbol{\Lambda}, \tag{4}$$

*For convenience, it is assumed that all the feature vectors are centered in the feature space $\mathcal{F}$ using $\tilde{\phi}(\boldsymbol{x}) := \phi(\boldsymbol{x}) - \frac{1}{N}\sum_{i=1}^N \phi(\boldsymbol{x}_i)$. Otherwise, the kernel matrix is first centered [22].



(a) Gen-RKM (b) Robust Gen-RKM

Figure 2: Illustration of robustness against outliers on the MNIST dataset. 20 % of the training data is contaminated with noise. The (robust) Gen-RKM is trained with a 2-dimensional latent space in the standard setup, see Section 4. The figure on the left shows the latent space of Gen-RKM, the figure on the right the robust version. The presence of outliers distorts the distribution of the latent variables in the case of the Gen-RKM, where the histogram of the latent variables is skewed. By downweighting the outliers (the outliers are put to zero), the histogram resembles a continuous Gaussian distribution again.

where $\boldsymbol{H} = \left[\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\right] \in \mathbb{R}^{s \times N}$ with $s \leq N$ is the number of selected principal components and $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is the kernel matrix corresponding to the input data. Using Mercer's theorem [25], positive-definite kernel functions $k : \mathbb{R}^{d_f} \times \mathbb{R}^{d_f} \mapsto \mathbb{R}$ are defined such elements of the kernel matrix are $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$. The feature map $\phi(\cdot)$ is implicitly defined by the kernel function and maps the input data to the (possibly infinite) feature space. 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 Laplace kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2/\sigma}$ [26]. One can also define explicit feature maps (e.g. by a neural network), where the positive-definiteness of the kernel function is preserved by construction [17].

2.2 Weighted Conjugate Feature Duality

We extend the notation of conjugate feature duality with a weighting matrix. Let $\boldsymbol{D} \succ 0$ be a symmetric positive-definite weighting matrix, then the following holds for any two vectors $\boldsymbol{e}, \boldsymbol{h}$:

$$\frac{1}{2\lambda} \boldsymbol{e}^\top \boldsymbol{D} \boldsymbol{e} + \frac{\lambda}{2} \boldsymbol{h}^\top \boldsymbol{D}^{-1} \boldsymbol{h} \geq \boldsymbol{e}^\top \boldsymbol{h}. \tag{5}$$

The inequality could be verified using the Schur complement by writing the above in its quadratic form:

$$\frac{1}{2} \begin{bmatrix} \boldsymbol{e}^\top & \boldsymbol{h}^\top \end{bmatrix} \begin{bmatrix} \frac{1}{\lambda}\boldsymbol{D}\boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \lambda \boldsymbol{D}^{-1}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{h} \end{bmatrix} \geq 0. \tag{6}$$

It states that for a matrix $\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^\top & \boldsymbol{C} \end{bmatrix}$, one has $\boldsymbol{Q} \succeq 0$ if and only if $\boldsymbol{A} \succ 0$ and the Schur complement $\boldsymbol{C} - \boldsymbol{B}^\top \boldsymbol{A}^{-1} \boldsymbol{B} \succeq 0$ [27], which proves the above inequality. The weighted kernel PCA objective in the LS-SVM setting is given by [28]:

$$\begin{aligned} \min_{\boldsymbol{U}, \boldsymbol{e}} J &= \frac{\eta}{2} \operatorname{Tr}(\boldsymbol{U}^\top \boldsymbol{U}) - \frac{1}{2\lambda} \boldsymbol{e}^\top \boldsymbol{D} \boldsymbol{e} \\ &\text{s.t. } \boldsymbol{e}_i = \boldsymbol{U}^\top \phi(\boldsymbol{x}_i), \quad \forall i = 1, \ldots, N. \end{aligned} \tag{7}$$

Combing the weighted conjugate feature duality with the above equation, we get:

$$\begin{aligned} J &\leq \frac{\eta}{2} \operatorname{Tr}(\boldsymbol{U}^\top \boldsymbol{U}) - \sum_{i=1}^{N} \left(\boldsymbol{e}_i^\top \boldsymbol{h}_i - \frac{\lambda}{2} \boldsymbol{D}_{ii}^{-1} \boldsymbol{h}_i^\top \boldsymbol{h}_i \right) \\ &\leq \sum_{i=1}^{N} \left(-\boldsymbol{e}_i^\top \boldsymbol{h}_i + \frac{\lambda}{2} \boldsymbol{D}_{ii}^{-1} \boldsymbol{h}_i^\top \boldsymbol{h}_i \right) + \frac{\eta}{2} \operatorname{Tr}(\boldsymbol{U}^\top \boldsymbol{U}), \end{aligned} \tag{8}$$

which gives the weighted kernel PCA objective function in the RKM setting:

$$\mathcal{J}_t^{\boldsymbol{D}} = \sum_{i=1}^{N} \left(-\phi(\boldsymbol{x}_i)^\top \boldsymbol{U}\boldsymbol{h}_i + \frac{\lambda}{2}\boldsymbol{D}_{ii}^{-1}\boldsymbol{h}_i^\top \boldsymbol{h}_i \right) + \frac{\eta}{2}\operatorname{Tr}(\boldsymbol{U}^\top \boldsymbol{U}). \tag{9}$$

The stationary points of $\mathcal{J}_t^{\boldsymbol{D}}$ are given by:

$$\begin{cases} \frac{\partial \mathcal{J}_t^{\boldsymbol{D}}}{\partial \boldsymbol{h}_i} = 0 \implies \lambda \boldsymbol{D}_{ii}^{-1}\boldsymbol{h}_i = \boldsymbol{U}^\top \phi(\boldsymbol{x}_i), \ \forall i = 1, \ldots, N \\ \frac{\partial \mathcal{J}_t^{\boldsymbol{D}}}{\partial \boldsymbol{U}} = 0 \implies \boldsymbol{U} = \frac{1}{\eta}\sum_{i=1}^{N} \phi(\boldsymbol{x}_i)\boldsymbol{h}_i^\top. \end{cases} \tag{10}$$

Eliminating $\boldsymbol{U}$ and denoting $\boldsymbol{H} \coloneqq [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N]$, $\boldsymbol{\Lambda} \coloneqq \operatorname{diag}\{\lambda_1, \ldots, \lambda_s\} \in \mathbb{R}^{s \times s}$ with s the dimension of the latent space, we get the following weighted eigenvalue problem:

$$\frac{1}{\eta}\boldsymbol{D}\boldsymbol{K}\boldsymbol{H}^\top = \boldsymbol{H}^\top \boldsymbol{\Lambda}. \tag{11}$$

2.3 Generation

Similarly as in the work of [18, 20], RKMs can be used in a generative setting. Given the learned interconnection matrix $\boldsymbol{U}$, and a given latent variable $\boldsymbol{h}^\star$, consider the following objective function:

$$\mathcal{J}_g = -\phi(\boldsymbol{x}^\star)^\top \boldsymbol{U}\boldsymbol{h}^\star + \frac{1}{2}\phi(\boldsymbol{x}^\star)^\top \phi(\boldsymbol{x}^\star), \tag{12}$$

with an additional regularization term on the input data. Here $\mathcal{J}_g$ denotes the objective function for generation. To reconstruct or denoise a training point, $\boldsymbol{h}^\star$ is equal to the corresponding hidden unit of the training point. Random generation is done by fitting a distribution on the learned latent variables, afterwards sampling a random $\boldsymbol{h}^\star$. Solving the stationary points of equation 12, the generated feature vector is equal to [20, 18]:

$$\phi(\boldsymbol{x}^\star) = \left(\frac{1}{\eta}\sum_{i=1}^{N} \phi(\boldsymbol{x}_i)\boldsymbol{h}_i^\top \right) \boldsymbol{h}^\star. \tag{13}$$

To obtain the generated data in the input space, the inverse image of the feature map $\phi(\cdot)$ is computed. This problem 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 $(\psi \circ \phi)(\boldsymbol{x}^\star) \approx \boldsymbol{x}^\star$, where $\phi(\boldsymbol{x}^\star)$ is calculated using equation 13. When using kernel methods, explicit feature maps are not necessarily known. The pre-image problem is known to be ill-conditioned [29]. An overview of different pre-image methods can be found in [30]. A second approach is to explicitly define pre-image maps and learn the parameters in the training procedure, as shown in [20]. In the experiments, we use a set of (convolutional) neural networks as the feature maps $\phi_{\boldsymbol{\theta}}(\cdot)$. Another (transposed convolutional) neural network is used for the pre-image map $\psi_{\boldsymbol{\zeta}}(\cdot)$ [31]. The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\zeta}$ correspond to the network weights. These weights are learned by minimizing the reconstruction error in combination with the weighted RKM objective function. In Section 3.2, the complete training algorithm is described.

To reconstruct or denoise an out-of-sample test point $\boldsymbol{x}^\star$, the data is projected on the latent space using:

$$\boldsymbol{h}^\star = \lambda^{-1}\boldsymbol{U}^\top \phi(\boldsymbol{x}^\star) = \frac{1}{\lambda\eta}\sum_{i=1}^{N} \boldsymbol{h}_i k(\boldsymbol{x}_i, \boldsymbol{x}^\star). \tag{14}$$

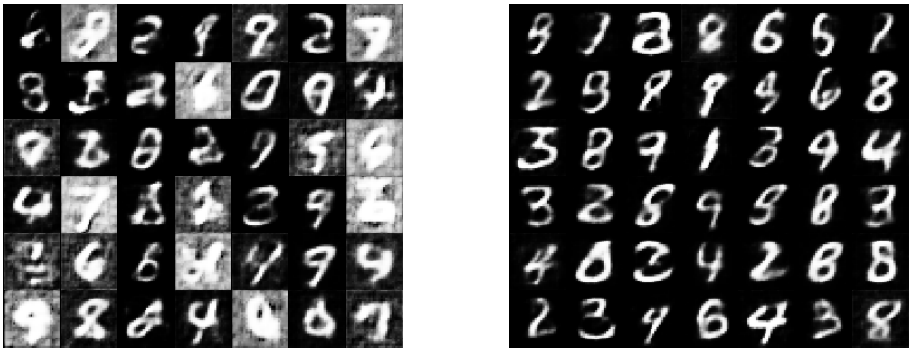
Afterwards the denoised point is reconstructed by projecting back to the input space using equation 13 followed by the pre-image map $\boldsymbol{\theta}$ and $\boldsymbol{\zeta}$.

3 Robust estimation of the latent variables

3.1 Robust Weighting Scheme

The weighted kernel PCA puts extra constraints on the hidden variables without losing the orthogonality of the eigendecomposition. In this paper, 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 $\boldsymbol{D}_{ii}$ corresponding to every $\boldsymbol{h}_i$:

$$\boldsymbol{D}_{ii} = \begin{cases} 1 & \text{if } d_i^2 \leq \chi^2_{s,\alpha} \\ 10^{-4} & \text{otherwise} \end{cases} \tag{15}$$



(a) Gen-RKM (b) Robust Gen-RKM

Figure 3: 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.

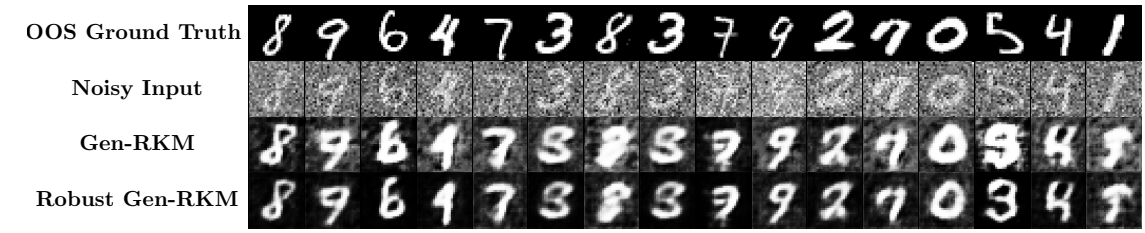


Figure 4: Illustration of robust denoising on the MNIST dataset. 20 % of the training data is contaminated with noise. The first and second row show the clean and noisy test images respectively. The third and fourth row show the denoised image using the Gen-RKM and robust Gen-RKM.

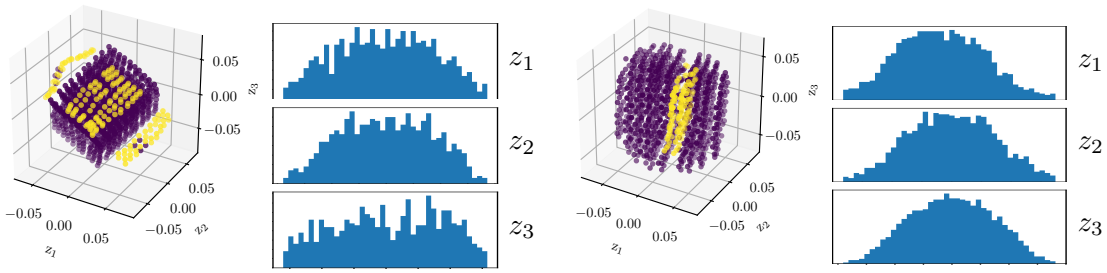
with s the dimension of the latent space, α the significance level of the Chi-squared distribution and d_i^2 is the Mahalanobis distance for the corresponding $\boldsymbol{h}_i$:

$$d_i^2 = (\boldsymbol{h}_i - \hat{\boldsymbol{\mu}})^\top \hat{\boldsymbol{S}}^{-1} (\boldsymbol{h}_i - \hat{\boldsymbol{\mu}}), \tag{16}$$

with $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{S}}$ the robustly estimated mean and covariance matrix respectively. In this paper, we propose to use the Minimum Covariance Determinant (MCD) [21]. The MCD is a highly robust estimator of multivariate location and scatter which has been used in many robust multivariate statistical methods [32, 33, 34]. 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 [35] 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 [36]. The same holds in the Gen-RKM framework, where the latent variables are maximized around zero. 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 equation 3), which implicitly puts a Gaussian prior on the hidden units. When the training of feature map is done correctly, one expects the latent variables to be normally distributed around zero [17]. Gaussian distributed latent variables are essential for having a *continuous* and *smooth* latent space, allowing easy interpolation. This property was studied in VAEs [5], where an extra 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 introduced. When training the Gen-RKM in the presence of outliers, the contamination can severely distort the distribution of the latent variables. This effect is seen in Figure 2, where a discontinuous and skewed distribution is visible. In the illustration, Gen-RKM and the robust counterpart where trained on the contaminated MNIST dataset using a 2-dimensional latent space for easy visualization. The rest of the parameters are the standard setup that is described in Section 4.

The role of Gaussian distributed latent variables was already studied in several other works, where multiple links with disentanglement are made. In β-VAEs [8], an adjustable hyperparameter β is introduced that balances latent channel capacity and independence constraints with reconstruction accuracy. The choice of parameter $\beta = 1$ corresponds to the original VAE formula-



(a) Gen-RKM (b) Robust Gen-RKM

Figure 5: Illustration of disentanglement on the DSprites 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 downweighted in the robust Gen-RKM, which moves them to the center of the scatter plot.

tion. In [8], they show that with $\beta > 1$ (more emphasis on the latent variables to be Gaussian distributed) the model is capable of learning a more efficient latent representation of the data, which is disentangled if the data contains at least some underlying factors of variation that are independent. In [37], the effect of the β term is analyzed more in depth. It was suggested that the stronger pressure for the posterior to match the factorised unit Gaussian prior puts extra constraints on the implicit capacity of the latent bottleneck [8]. Chen et al. [38] show a decomposition of the variational lower bound that can be used to explain the success of the β-VAE [8] 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. In the (robust) Gen-RKM formulation, the latent variables are already uncorrelated by construction due to the eigendecomposition.

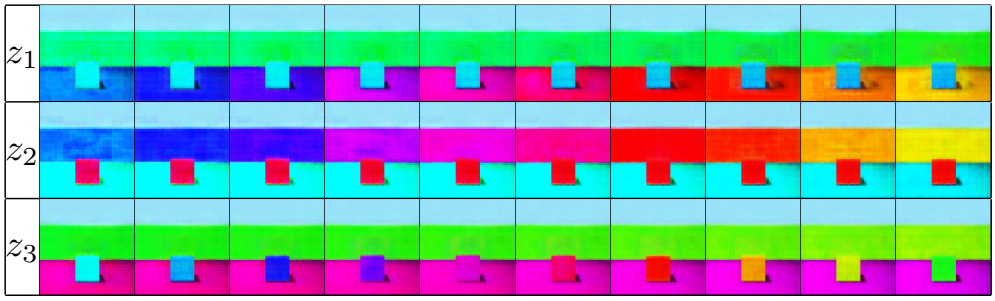


Figure 6: Illustration of latent traversals along the 3 latent dimensions for 3DShapes dataset using the robust Gen-RKM model. The first, second and third row distinctly captures the floor-hue, wall-hue and object-hue respectively while keeping other generative factors constant. Table 2 shows the disentanglement metrics for standard and robust cases.

3.2 Algorithm

We propose to use the above described reweighting step within the Gen-RKM framework [20]. 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 [20] with an extra re-weighting step wherein the system in equation 11 is solved. Furthermore, the reconstruction error is weighted to reduce the effect of potential outliers. The loss function now becomes:

$$\min_{\boldsymbol{\theta},\boldsymbol{\zeta}} \mathcal{J}_c^{\boldsymbol{D}} = \mathcal{J}_t^{\boldsymbol{D}} + \frac{c_{\text{stab}}}{2}(\mathcal{J}_t^{\boldsymbol{D}})^2 + \frac{c_{\text{acc}}}{N}\sum_{i=1}^{N} \boldsymbol{D}_{ii}\mathcal{L}(\boldsymbol{x}_i, \psi_{\boldsymbol{\zeta}}(\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i))), \tag{17}$$

where $c_{stab} \in \mathbb{R}^+$ is a stability constant [16] 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 [20]. The adapted training algorithm is given in the Appendix.

4 Experiments

In this section, we evaluate the robustness of the weighted Gen-RKM on the MNIST [39], Fashion-MNIST [40], Dsprites [41] and 3D shapes [42] dataset. Training of the robust Gen-RKM is done using the algorithm proposed in Section 3.2, where we take $N_{\mathrm{MCD}} = \lfloor N \times 0.75 \rfloor$ and $\alpha = 0.975$ for the Chi-squared distribution (see equation 15). Afterwards we compare with the standard Gen-RKM [20]. The models have the same encoder/decoder architecture, optimization parameters and are trained until convergence. Information on the datasets and model architectures is given in the Appendix.

4.1 Generation and Denoising

Figure 3 shows the generation of random images using Gen-RKM and the robust counterpart 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. 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 considerably better. This is confirmed by the Fréchet Inception Distance (FID) scores [43] in Table 1, which quantify the quality of generation. We repeat the above experiment on the Fashion-MNIST dataset, where only the FID scores are reported in Table 1. Next, we use the generative models in a 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 image are retained while insignificant features like noise are removed. Figure 4 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 $\mathrm{MAE} = 0.415$ is much higher than the robust version $\mathrm{MAE} = 0.206$.

The experiments show that basic Gen-RKM is highly affected by outliers, while the robust counterpart can cope with a significant fraction of contamination.

Table 1: FID Scores [43] for 3000 randomly generated samples when the training data is contaminated with 20 % outliers. (smaller is better).

Dataset	**FID score**	
MNIST	Gen-RKM	Robust Gen-RKM
	135.95	**87.03**
Fashion-MNIST	Gen-RKM	Robust Gen-RKM
	163.70	**155.32**

4.2 Effect on Disentanglement

In the previous section, the datasets are contaminated by adding random noise. In this experiment, contamination is an extra generating factor which is not present for 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 minimal scale and 0 degree orientation respectively. Afterwards, the training data is contaminated by cylinders with maximal scale at 30 degree orientation (20 % of the data is considered as outliers). 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. Figure 5 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 downweighted. An illustration of latent traversals along the 3 latent dimensions using the robust Gen-RKM model is given in Figure 6, where the robust model is capable of disentangling the 3 'clean' generating factors.

The disentanglement of the latent representation is measured quantitatively using the proposed framework[†] of [44], which consists out of 3 measures: disentanglement, completeness and informativeness. The results are shown in Table 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 datapoints with minimal scale and 0 degree 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 (20 % of the data is considered as outliers). 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 factor (orientation, scale and shape) which only occur in a small fraction. We only show the quantitative results in Table 2.

[†]Code and dataset available at `https://github.com/cianeastwood/qedr`

Table 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 [44] with Lasso and Random Forest regressor [44] is used to evaluate the learned representation. For disentanglement and completeness higher score is better, for informativeness, lower is better.

Dataset			**Lasso**			**Random Forest**		
	h_{dim}	**Algorithm**	**Disent.**	**Comple.**	**Inform.**	**Disent.**	**Comple.**	**Inform.**
DSprites	2	Gen-RKM	0.07	0.07	**5.82**	0.25	0.27	**5.91**
		Robust Gen-RKM	**0.21**	**0.21**	9.13	**0.36**	**0.38**	5.95
3D Shapes	3	Gen-RKM	0.14	0.14	**3.03**	0.15	0.15	1.09
		Robust Gen-RKM	**0.47**	**0.49**	3.13	**0.44**	**0.45**	**1.02**

5 Conclusion

Using a weighted conjugate feature duality, a 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 inspite of contamination in training data. Furthermore, being a latent variable model, robust Gen-RKM preserves the disentangled representation. Future works consists of exploring various weighting schemes to control the effect of sampling bias in the data and other robust estimators.

Acknowledgment

EU: The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. Research Council KUL: Optimization frameworks for deep kernel machines C14/18/068 Flemish Government: FWO: projects: GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), PhD/Postdoc grant Impulsfonds AI: VR 2019 2203 DOC.0318/1QUATER Kenniscentrum Data en Maatschappij Ford KU Leuven Research Alliance Project KUL0076 (Stability analysis and performance improvement of deep reinforcement learning algorithms). The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government department EWI.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–2680.

[2] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 727–736.

[3] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th ICML*, 2008, pp. 1096–1103.

[4] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proceedings of the 24th ICML*, 2007, pp. 791–798.

[5] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[6] P. Smolensky, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281.

[7] R. Salakhutdinov and G. Hinton, "Deep Boltzmann Machines," *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, vol. Volume 5 of JMLR, 2009.

[8] 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." *ICLR*, vol. 2, no. 5, p. 6, 2017.

[9] 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*, 2016, pp. 2172–2180.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017, pp. 214–223.

[12] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, 2016, pp. 469–477.

[13] F. Futami, I. Sato, and M. Sugiyama, "Variational inference based on robust divergences," *21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[14] T. Kaneko, Y. Ushiku, and T. Harada, "Label-noise robust generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2467–2476.

[15] G. G. Chrysos, J. Kossaifi, and S. Zafeiriou, "Robust conditional generative adversarial networks," *International Conference on Learning Representations (ICLR*, 2019.

[16] J. A. K. Suykens, "Deep Restricted Kernel Machines using Conjugate Feature Duality," *Neural Computation*, vol. 29, no. 8, pp. 2123–2163, Aug. 2017.

[17] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. River Edge, NJ: World Scientific, Jan. 2002.

[18] J. Schreurs and J. A. K. Suykens, "Generative Kernel PCA," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)* , 2018, pp. 129–134.

[19] L. Houthuys and J. A. K. Suykens, "Tensor learning in multi-view kernel PCA ," in *27th International Conference on Artificial Neural Networks ICANN, Rhodes, Greece*, vol. 11140, 2018, pp. 205–215.

[20] A. Pandey, J. Schreurs, and J. A. K. Suykens, "Generative restricted kernel machines," *arXiv preprint arXiv:1906.08144*, 2019.

[21] P. J. Rousseeuw and K. V. Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[22] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *International conference on artificial neural networks*. Springer, 1997, pp. 583–588.

[23] R. T. Rockafellar, *Conjugate Duality and Optimization*. SIAM, 1974.

[24] 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.*, vol. 2, 2004, pp. II–97–104 Vol.2.

[25] 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*, vol. 209, no. 441-458, pp. 415–446, Jan. 1909.

[26] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[28] C. Alzate and J. A. Suykens, "Multiway spectral clustering with out-of-sample extensions through weighted kernel pca," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 2, pp. 335–347, 2008.

[29] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and De-noising in Feature Spaces," in *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*. MIT Press, 1999, pp. 536–542.

[30] P. Honeine and C. Richard, "Preimage Problem in Kernel-Based Machine Learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 77–88, Mar. 2011.

[31] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[32] M. Hubert, P. J. Rousseeuw, and K. Vanden Branden, "ROBPCA: a new approach to robust principal component analysis," *Technometrics*, vol. 47, no. 1, pp. 64–79, 2005.

[33] S. Engelen, M. Hubert, K. V. Branden, and S. Verboven, "Robust PCR and Robust PLSR: a comparative study," in *Theory and applications of recent robust methods*. Springer, 2004, pp. 105–117.

[34] M. Hubert, P. J. Rousseeuw, and T. Verdonck, "A deterministic algorithm for robust location and scatter," *Journal of Computational and Graphical Statistics*, vol. 21, no. 3, pp. 618–637, 2012.

[35] C. Croux and G. Haesbroeck, "Influence function and efficiency of the minimum covariance determinant scatter matrix estimator," *Journal of Multivariate Analysis*, vol. 71, no. 2, pp. 161–190, 1999.

[36] 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*, vol. 14, no. 2, pp. 447–450, 2003.

[37] 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.

[38] 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*, 2018, pp. 2610–2620.

[39] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[40] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," 2017.

[41] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, "dsprites: Disentanglement testing sprites dataset," https://github.com/deepmind/dsprites-dataset/, 2017.

[42] C. Burgess and H. Kim, "3d shapes dataset," https://github.com/deepmind/3dshapes-dataset/, 2018.

[43] 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 *Proceedings of the 31st NIPS*, 2017, pp. 6629–6640.

[44] C. Eastwood and C. K. I. Williams, "A framework for the quantitative evaluation of disentangled representations," in *International Conference on Learning Representations (ICLR)*, 2018.

See Table 4 and Table 3 for details on model architectures, datasets and hyperparameters used in this paper. The PyTorch library in Python was used as the programming language witha 8GB NVIDIA QUADRO P4000 GPU

Table 3: Model architectures. 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 has sigmoid activation function.

Dataset	**Optimizer** (Adam)		**Architecture** $\mathcal{X}$
MNIST	1e-4	Input	28x28x1
		Feature-map (fm)	Conv 32x4x4; Conv 64x4x4;
		Pre-image map	FC 228 (Linear) reverse of fm
		Latent space dim.	10
Fashion	1e-4	Input	28x28x1
		Feature-map (fm)	Conv 32x4x4; Conv 64x4x4;
		Pre-image map	FC 228 (Linear) reverse of fm
		Latent space dim.	10
Dsprites	1e-4	Input	64x64x1
		Feature-map (fm)	Conv 20x4x4; Conv 40x4x4; Conv 80x4x4;
		Pre-image map	FC 228 (Linear) reverse of fm
		Latent space dim.	2
3D Shapes	1e-4	Input	64x64x3
		Feature-map (fm)	Conv 30x4x4; Conv 60x4x4; Conv 90x4x4;
		Pre-image map	FC 228 (Linear) reverse of fm
		Latent space dim.	3

Table 4: Datasets and hyperparameters used for the experiments. N is the dataset size, d the dimension, N_{subset} is the size of the subset used for training, s the latent space dimension and m the minibatch size.

Dataset	N	d	N_{subset}	s	m
MNIST	60000	28×28	3000	10	200
Fashion-MNIST	60000	28×28	3000	10	200
Dsprites	737280	64×64	1024	2	200
3D Shapes	480000	$64 \times 64 \times 3$	1200	3	200

Algorithm 1 Weighted Gen-RKM: training

Input: $\{\boldsymbol{x}_i\}_{i=1}^N,\ \eta$, feature map $\phi(\cdot)$ - explicit *or* implicit via kernels $k(\cdot,\cdot)$
Output: Feature map parameters $\boldsymbol{\theta}, \boldsymbol{\zeta}$ and hidden units $\{\boldsymbol{h}_i\}_{i=1}^N$

procedure TRAIN
  if $\phi_j(\cdot)$ = Implicit **then**
    Hyperparameters: kernel specific
    Solve Eq. equation 4
    Select s principal components
    Determine the weights using Eq. equation 15
    Re-estimate the hidden units using Eq. equation 11
  else if $\phi_j(\cdot)$ = Explicit **then**
    while not converged **do**
      $\{\boldsymbol{x}\} \leftarrow \{$Get mini-batch$\}$
      $\phi(\boldsymbol{x}) \leftarrow \boldsymbol{x}$
      do steps 4-7
      $\phi(\boldsymbol{x}) \leftarrow h$ Eq. equation 13
      $\boldsymbol{x} \leftarrow \psi(\phi(\boldsymbol{x}))$
      $\Delta\boldsymbol{\theta} \propto -\nabla_{\boldsymbol{\theta}} \mathcal{J}_t^{\boldsymbol{D}}$
      $\Delta\boldsymbol{\zeta} \propto -\nabla_{\boldsymbol{\zeta}} \mathcal{J}_t^{\boldsymbol{D}}$
    end while
  end if
end procedure