KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering

Incorporation of Prior Knowledge into Kernel Based Models

Siamak Mehrkanoon

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

July 2015

Incorporation of Prior Knowledge into Kernel Based Models

Siamak MEHRKANOON

Jury:
Prof. dr. ir. Adhemar Bultheel, chair
Prof. dr. ir. Johan A. K. Suykens, promotor
Prof. dr. ir. Stefan Vandewalle
Prof. dr. ir. Moritz Diehl
Prof. dr. ir. Edwin Reynders
Prof. dr. Roland Toth
(Eindhoven University of Technology)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

July 2015

© 2015 KU Leuven – Faculty of Engineering
Uitgegeven in eigen beheer, Siamak Mehrkanoon, Kasteelpark Arenberg 10 bus 2446, B-3001 Heverlee (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.

To

my dear parents, Jafar and Nilofar, for their unconditional support

and

my darling wife Noushin for her great love

Foreword

I would like to take this opportunity to acknowledge all those who made this thesis possible. First and foremost, I truly want to express my sincere and deepest gratitude to Prof. Johan Suykens, my promotor, who gave me the opportunity to join his lab at KU Leuven and guided my research. Johan, I am very grateful of your continuous support, daily feedback, valuable suggestions, fruitful discussions and for inspiring many of the key ideas that are presented in this thesis. Special acknowledgments are extended to Prof. Stefan Vandewalle, Prof. Moritz Diehl, Prof. Edwin Reynders and Prof. Roland Toth for being part of the jury of this thesis and providing valuable comments which helped me to improve the concept of the thesis.

I would like to thank my colleagues and friends at ESAT with whom I spent the most memorable moments and I collaborated on several occasions. I have enjoyed the scientific life at ESAT and the memories made at both old and new buildings will remain with me forever. Not to forget the best time of the day and our great discussions formed during lunchtime at ALMA with all the lab members. I will never forget the time that I have spent with my old and new lunch buddies: Carlos, Carolina, Dries, Philippe, Kim, Maarten, Marco, Marko, Mauricio, Lynn, Raghvendra, Vilen, Emanuele, Ricardo, Antoine, Michaël, Bertrand, Hanyuan Gervasio, Yuning, Yunlong and others. Marin and Tena I have not forgotten you, the little Vita and the time we shared together at Oude Heverlee! I would like to extend my special thanks to Lynn for improving my basic knowledge of Dutch and translating the thesis abstract.

Great thanks to my friend, Xiaolin who made my journey to China a truly wonderful experience, there I had the opportunity to meet with Prof. Wang and Prof. Lili with whom I enjoyed the taste of authentic Chinese cuisine in addition to scientific discussions. I am grateful of the friends I made at Tsinghua University: Kuang Yu, Jing Wang, Juntang Yu and Xiangming Xi for their warm welcome and hospitality during my stay in Beijing. I would like to extend my appreciation to the whole administrative staffs at KU Leuven,

ESAT-STADIUS, specially Ida Tassens and John Vos for the help I received from them during my PhD study years.

My endless thanks and appreciations go to my family, my compassionate father Jafar, and my gracious mother Nilofar and my sympathetic wife Noushin, for their love, patience and encouragement. I'll never be able to express enough appreciation to my parents, whom their incredible love, hard work and sincere hopes for my success, were my main sources of inspiration and support throughout the life. I owe so much to my beloved wife for her understanding, care and companionship. I wish to thank her from the bottom of my heart for all the strength and hope she gave me, while kindly accompanied me throughout my postgraduate years. To my family I owe all that I am and all that I have ever fulfilled. Last but not least, my special thanks go to my elder brother Saeid with whom I enjoyed a lot discussing and collaborating on several occasions.

Siamak Mehrkanoon
Leuven, July 2015.

Abstract

Incorporation of the available prior knowledge into to learning framework can play an important role in improving the generalization of a machine learning algorithm. The type of available side information can vary depending on the context. The scope of this thesis is the development of learning algorithms that exploit the side information. In particular the focus has been on learning the solution of a dynamical system, parameter estimation and semi-supervised learning. To this end, the prior knowledge is incorporated into the kernel based core model via adding a regularization term and/or set of constraints.

In the context of dynamical systems, the available differential equations together with initial/boundary conditions are considered as side information. Starting from a least squares support vector machines (LSSVM) core formulation, the extension to learn the solution of dynamical system governed by ordinary differential equations (ODEs), differential algebraic equations (DAEs) and partial differential equations (PDEs) are considered. The primal-dual optimization formulation typical of LSSVM allows the integration of side information by modifying the primal problem.

A kernel based approach for estimating the unknown (constant/time-varying) parameters of a dynamical system described by ordinary differential equations (ODEs) is introduced. The LSSVM serves as a core model to estimate the state trajectories and its derivatives based on the observational data. The approach presents a number of advantages. In particular, it avoids repeated integration of the system and also in case of parameter affine systems, one obtains a convex optimization problem. Moreover for systems with delays (state delay), where the objective function can be non-smooth, the approach shows promising results by converting the problem into an algebraic optimization problem.

In many applications ranging from machine learning to data mining, obtaining the labeled samples is costly and time consuming. On the other hand with the recent development of information technologies one can easily encounter a

huge amount of unlabeled data coming from the web, smartphones, satellites etc. In these situations, one may consider to design an algorithm that can learn from both labeled and unlabeled data. In this context, elements such as dealing with data streams (real time data analysis), scalability to large-scale data and model selection criteria become key aspects. Starting from the Kernel Spectral Clustering (KSC) core formulation, which is an unsupervised algorithm, extensions towards integration of available side information and devising a semi-supervised algorithm are a scope of this thesis. A novel multi-class semi-supervised learning algorithm (MSS-KSC) is developed that addresses both semi-supervised classification and clustering. The labeled data points are incorporated into the KSC formulation at the primal level via adding a regularization term. This converts the solution of KSC from an eigenvalue problem to a system of linear equations in the dual. The algorithm realizes a low dimensional embedding for discovering micro clusters.

Though the portion of labeled data points is small, one can easily encounter a huge amount of the unlabeled data points. In order to make the algorithm scalable to large scale data two approaches are proposed, Fixed-size and reduced kernel MSS-KSC (FS-MSS-KSC and RD-MSS-KSC). The former relies on the Nyström method for approximating the feature map and solves the problem in the primal whereas the latter uses a reduced kernel technique and solves the problem in the dual. Both approaches possess the out-of-sample extension property to unseen data points.

In today's applications, evolving data streams are ubiquitous. Due to the complex underlying dynamics and non-stationary behavior of real-life data, the demand for adaptive learning mechanisms is increasing. An incremental multi-class semi-supervised kernel spectral clustering (I-MSS-KSC) algorithm is proposed for an on-line clustering/classification of time-evolving data. It uses the available side information to continuously adapt the initial MSS-KSC model and learn the underlying complex dynamics of the data stream. The performance of the proposed method is demonstrated on synthetic data sets and real-life videos. Furthermore, for the video segmentation tasks, Kalman filtering is used to provide the labels for the objects in motion and thereby regularizing the solution of I-MSS-KSC.

Abbreviations

ARI	Adjusted Rand Index
BVP	Boundary Value Problem
BLF	Balanced Line Fit
DAE	Differential Algebraic Equation
DDE	Delay Differential Equations
FS-MSS-KSC	Fixed Size Multiclass Semi-Supervised Kernel Spectral Clustering
IKM	Incremental K-means
I-MSS-KSC	Incremental Multiclass Semi-Supervised Kernel Spectral Clustering
IVP	Initial Value problem
KKT	Karush-Kuhn-Tucker
KSC	Kernel Spectral Clustering
LapSVMp	Laplacian Support Vector Machines in primal
LSSVM	Least Squares Support Vector Machine
MSS-KSC	Multiclass Semi-Supervised Kernel Spectral Clustering
NLP	Nonlinear Programming Problem
NMI	Normalized Mutual Information
ODE	Ordinary Differential Equation
PCA	Principal Component Analysis
PDE	Partial Differential Equation
PEM	Prediction Error Method
RBF	Radial Basis Function
RD-MSS-KSC	Reduced Multiclass Semi-Supervised Kernel Spectral Clustering
SC	Spectral Clustering
SVM	Support Vector Machine

Notation

x^T	Transpose of a vector x
A^T	Transpose of a matrix A
A_{ij}	ij-th entry of the matrix A
I_N	$N \times N$ Identity matrix
1_N	$N \times 1$ Vector of ones
$\varphi(\cdot)$	Feature map
Φ	Feature matrix
$K(x_i, x_j)$	Kernel function evaluated on data points x_i, x_j
$\lvert\mathcal{S}\rvert$	Cardinality of a set $\mathcal{S}$
$A(:, i)$	Matlab notation for the i-th column of matrix A
$A(i, :)$	Matlab notation for the i-th row of matrix A
$A(k : l, m : n)$	submatrix of matrix A consisting of rows k to l and columns m to n
$\frac{\partial^2 f}{\partial x^2}$	Second order partial derivative of f w.r.t x
$\min_x f(x)$	Minimization over x, minimal function value returned
$\operatorname{argmin}_x f(x)$	Minimization over x, optimal value of x returned

Contents

Abstract v

Abbreviations vii

Notation ix

Contents xi

1 Introduction 1

1.1 General Background 1

1.2 Challenges 2

1.3 Methodology 3

1.4 Objectives 4

1.5 Overview of Chapters 6

1.6 Contributions of the Thesis 10

2 Learning and Kernel Based Models 15

2.1 Kernel Methods 15

2.2 Least Squares Support Vector Machines 16

2.2.1 Regression Problem 16

2.2.2 Classification Problem 17

2.3 Kernel Spectral Clustering 18

3 Learning Solutions of Dynamical Systems 23

3.1 Related Work 23
3.2 Learning the Solution of ODEs 27
3.2.1 Problem statement and overview of existing methods . . 28
3.2.2 Definitions of some operators 29
3.2.3 Formulation of the method for first order IVP 31
3.2.4 Formulation of the method for second order IVP and BVP 33
3.2.5 Formulation of the method for the nonlinear ODE case 37
3.2.6 Solution on a long time interval 38
3.3 Learning the Solution of DAEs 39
3.3.1 Formulation of the method for IVPs in DAEs 40
3.3.1.1 Singular ODE System 41
3.3.1.2 Explicit ODE System 44
3.3.2 Formulation of the method for BVPs in DAEs 46
3.4 Learning the Solution of PDEs 50
3.5 Formulation of the Method 51
3.5.1 PDEs on rectangular domains 53
3.5.2 PDEs on irregular domains 57
3.5.3 Formulation of the method for nonlinear PDE 59
3.6 Model Selection 61
3.7 Experiments 62
3.7.1 ODE test problems 62
3.7.1.1 Sensitivity of the solution w.r.t the parameter 67
3.7.1.2 Large interval 68
3.7.2 DAE test problems 70

3.7.3 PDE test problems 74
3.8 Conclusions 81

4 Parameter Estimation of Dynamical Systems 83
4.1 Related Work 84
4.2 Dynamical Systems Governed by ODEs 86
4.2.1 Constant parameter estimation 86
4.2.2 Time varying parameter estimation 88
4.3 LSSVM Based Initialization Approach 90
4.4 Pre-processing using LSSVM 93
4.5 Dynamical Systems Governed by DDEs 93
4.5.1 Problem statement 94
4.5.1.1 Reconstruction of fixed delays 94
4.5.1.2 Reconstruction of time varying parameters . . 95
4.5.2 General Methodology 96
4.5.3 Fixed delay τ is unknown 96
4.5.4 Parameter $\theta(t)$ is unknown 100
4.5.5 History function $\mathcal{H}_1(t)$ is unknown 102
4.6 Experiments 103
4.6.1 Parameter estimation of ODEs 103
4.6.1.1 Constant parameters 103
4.6.1.2 Time varying parameters 108
4.6.2 Parameter estimation of DDEs 111
4.6.2.1 Constant parameters 112
4.6.2.2 Time varying parameters 115
4.7 Conclusions 118

5 Non-Parallel Classifiers 119

5.1 Related Work 119
5.2 Non-parallel Support Vector Machine 120
5.2.1 General formulation 121
5.2.2 Related existing methods 122
5.3 Different Loss Functions 123
5.3.1 Case: LS-LS loss 124
5.3.2 Case: LS-Hinge loss 128
5.3.3 Case: LS-Pinball loss 130
5.4 Guidelines for the User 132
5.5 Non-parallel Semi-Supervised KSC 133
5.5.1 Primal-Dual formulation of the method 134
5.6 Numerical Experiments 140
5.6.1 Classification 140
5.6.2 Semi-supervised classification 144
5.7 Conclusions 147

6 Semi-Supervised Learning 149

6.1 Related Work 150
6.2 Semi-Supervised Classification 152
6.2.1 Primal-Dual formulation of the method 153
6.2.2 Encoding/Decoding scheme 155
6.3 Semi-Supervised Clustering 156
6.3.1 From solution of linear systems to clusters: encoding . . 156
6.3.2 Low dimensional spectral embedding 157
6.4 Model Selection 158
6.5 Large Scale Semi-Supervised Learning 160
6.5.1 Approximation to the feature map 160

6.5.2 Fixed-Size MSS-KSC for large scale datasets 161
6.5.3 Subsample selection for Nyström approximation 163
6.5.4 Reduced MSS-KSC for large scale datasets 164
6.6 Experimental Results 168
6.6.1 Toy problems 168
6.6.2 Real-life benchmark data sets 169
6.6.3 Image segmentation 175
6.6.4 Community detection 179
6.6.5 Large scale data sets 181
6.7 Conclusions 186

7 Incremental Semi-Supervised Learning Regularized by Kalman Filtering 187

7.1 Related Work 188
7.2 Incremental Multi-class Semi-Supervised Clustering 189
7.2.1 Out-of-sample solution vector 190
7.2.2 Computational complexity 194
7.2.3 Regularizing I-MSS-KSC via Kalman filtering 194
7.3 Experimental Results 198
7.3.1 Synthetic data sets 200
7.3.2 Synthetic time-series 201
7.3.3 Real-life video segmentation 202
7.4 Conclusions 206

8 General Conclusions 215

8.1 Concluding Remarks 215
8.2 Future Research 217

A Appendix **219**

A.1 Symbolic Computing of LSSVM based models 219

A.2 Motivation 219

A.3 Development of Symbolic Solver 220

A.4 *SYM-LSSVM-SOLVER* Package 221

A.4.1 Procedure *Pro-Lag* 221

A.4.2 Procedure *Pro-KKT* 223

A.4.3 Procedure *Pro-Dual System* 226

A.4.4 Procedure *Pro_Dual Model* 227

A.5 GUI Application 227

A.6 Conclusion and future work 227

Bibliography **231**

Chapter 1

Introduction

1.1 General Background

Machine Learning is an actively growing research field that aims at extracting useful knowledge, unveiling hidden patterns and learning the underlying complex structure from data. Machine Learning has several connections with data mining, pattern recognition, statistics and optimization theory. Kernel methods are one of the successful branches in the fields of machine learning and data mining.

They can provide predictive models that often outperform competing approaches in terms of generalization performance. The main idea in kernel based methods is to map the data into a high dimensional space by means of a nonlinear feature map. A linear model in the feature space then corresponds to a nonlinear model in the original domain. Support Vector Machines (SVMs) proposed by Vapnik [167], is a well-known example of such a method, which has been successfully applied to non-linear classification and regression problems with high dimensional data.

In many practical applications, some forms of additional prior knowledge is often available. For instance in the context of nonlinear system identification, prior knowledge could be the applicability of a physical law for part of the system or information on its stability. In the context of clustering, one may know class labels for some items and letting them guide the clustering process. Incorporating available prior knowledge into the data driven modeling task can potentially improve the performance of the model. Therefore exploiting and incorporating the available prior information into the learning framework is

the scope of this thesis. In particular in this thesis the kernel based models cast in the Least Squares Support Vector Machines (LSSVM) framework [156], are considered as core models and the additional information is embodied in the models by adding regularization terms or sets of constraints to the primal optimization problem.

1.2 Challenges

Challenges tackled in this thesis are addressing the complications arising in the incorporation of prior knowledge into kernel based models for handling forward problems, inverse problems, classification/clustering and online learning.

- **Kernel based model towards learning the solution of a dynamical system:** Differential equations can be found in the mathematical formulation of physical phenomena in a wide variety of applications especially in science and engineering [45, 88]. Analytic solutions for these problems are not generally available and hence numerical methods must be applied. In the case of differential algebraic equations (DAEs), most of the existing methods are only applicable to low-index problems and often require the problem to have special structure. Furthermore, these approaches approximate the solutions at discrete points only (discrete solution) and some interpolation technique is needed in order to get a continuous solution. The challenge is to design a kernel based formulation for learning the solution of the given differential equations via incorporating the initial/boundary conditions into the core model. Addressing the model selection, performing simulation on long time intervals and dealing with nonlinear differential equations are additional challenges for using kernel based approaches in this context.
- **Parameter estimation of dynamical systems using kernel based model:** Parameter estimation of dynamical systems described by a set of differential equations is widely used in modelling of dynamic processes in physics, engineering and biology. The aim is to estimate the unknown parameters of the system based on the available observational data. In this thesis we consider parameter affine systems. Due to the nonlinear dynamics of the system, conventional approaches formulate the parameter estimation problem as a non-convex optimization problem. The challenge is to develop a kernel based method formulated as a convex optimization problem for estimating the unknown constant/time-varying parameters of the given dynamical system described by ordinary/delay differential equations. The approximated parameter then can serve as an initial guess

for the conventional approaches. Parameter estimation of a system with delays is a very important aspect in many applications and at the same time is a challenging problem as the objective function of the optimization problem for DDE might be non-smooth because the state trajectory might be non-smooth in the parameter and this will make the optimization problem more complicated.

- **Semi-supervised learning for realistic and large scale data size:** In many contexts, ranging from data mining to machine perception, obtaining the labels of input data is often difficult and expensive. Therefore in many cases one deals with a huge amount of unlabeled data, while the fraction of labeled data points will typically be small. In these cases one may consider to use a semi-supervised algorithm that can learn from both labeled and unlabeled data. The challenge is to devise a kernel-based model that is able to learn from few labeled data points and generalizes well on the unseen data points. In addition, in many applications ranging from text mining, information retrieval and computer vision the amount of (unlabeled) data points has been increasing at exponential rate. Therefore one also should take into account the scalability of the semi-supervised algorithms in order to deal with large scale data.
- **Online semi-supervised learning:** The behavior of a dynamic system can meet different regimes in the course of time i.e. the data distribution can change over time. In this case, in order to cope with non-stationary data-streams one needs to continuously adjust the model in order to better explain the whole dynamics of the underlying system. Considering that some labeled data points are available, a semi-supervised algorithm that can operate in an online fashion is desirable.

1.3 Methodology

This thesis explores the possibilities of incorporating the available side-information in the learning process. One can start with a suitable core model corresponding to the given task, and integrate the prior-knowledge of the task into the model via adding a set of constraints or regularization term. The general picture of the thesis is summarized in Fig. 1.1. The core model considered in this thesis are Least Squares Support Vector Machines (LSSVM) and Kernel Spectral Clustering (KSC). These are kernel based models and have been shown to be successful in many applications. They are formulated in the primal-dual setting and therefore one enjoys working with high-dimensional data by solving the problem in the dual. It should be mentioned that

among different ways to obtain a kernel based model such as following a probabilistic Bayesian setting or by using function estimation in a reproducing kernel Hilbert space, the primal-dual approach has the advantage that it is usually straightforward to incorporate additional structure or knowledge into the estimation problem. Thanks to the Nyström approximation method, one can also deal with large scale data (the number of data is much larger than the number of variables).

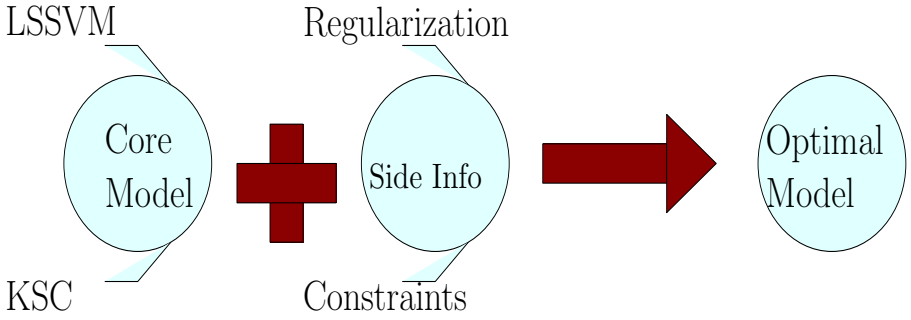


Figure 1.1: General picture of this thesis.

In the subsequent Chapters, it will be shown how one can learn the solution a dynamical system by adding a set of constraints to the LSSVM primal optimization problem. In the context of semi-supervised learning where one is interested to learn from a few labeled and a large amount of unlabeled data points, KSC is used as a core model. The labels are incorporated to the primal formulation of KSC by adding a regularization term. Adding this term changes the dual formulation from an eigenvalue problem to a system of linear equations but essential properties are maintained. Moreover, a different mechanism based on a Kalman filter to further regularize the solution of a developed semi-supervised learning algorithm is also discussed. This has some applications for instance in video segmentation task.

1.4 Objectives

The primary objective of this thesis is to explore the possibilities of incorporating prior knowledge into the learning framework which can result in improving the performance and achieving a richer model. In particular, the incorporation of side information into kernel based approaches in a range of application domains.

- **Kernel based framework for learning the solution of the dynamical systems:** From a kernel-based modeling point of view, one can consider the given differential equations together with its initial or boundary conditions as prior knowledge and seek the solution of the differential equation by means of Least Squares Support Vector Machines (LSSVMs) whose parameters are adjusted to minimize an appropriate error function. The problem is formulated as an optimization problem in the primal-dual setting. The approximate solution in the primal is expressed in terms of the feature map and is forced to satisfy the initial/boundary conditions using a constrained optimization problem. The optimal representation of the solution is then obtained in the dual. For the linear and nonlinear cases, these parameters are obtained by solving a system of linear and nonlinear equations, respectively. The method is well suited to solving mildly stiff, nonstiff, and singular ODEs with initial and boundary conditions. The solution of IVPs and BVPs in differential algebraic equations (DAES) with high index can also be learned without requiring to use any index-reduction technique.

- **Kernel based framework for parameter estimation of dynamical systems:** The parameter estimation is often formulated as a non-convex optimization problem and moreover repeated numerical integration of a given dynamical system is required. The objective here is to use LSSVM as core model and design a kernel-based method to formulate a convex optimization algorithm for parameter estimation of dynamical system in continuous time. Furthermore the developed method should not need repeated numerical integration. In addition we are interested to be able to estimate both constant and time-varying parameters of the system. In this thesis two types of differential equations i.e. ordinary and delay differential equations (ODEs and DDEs) are considered to describe the dynamics of the system.

- **Semi-supervised learning based on KSC core model:** Semi-supervised learning (SSL) is a framework in Machine Learning which aims at learning from both unlabeled and labeled data points. The aim here is to develop a multi-class semi-supervised learning algorithm that can address both semi-supervised classification and clustering. We aim at using a completely unsupervised algorithm as a core model so that the algorithm can learn from unlabeled data points. In addition, the side-information (labeled data points) is incorporated to the core model using a regularization term thus improving the model performance. The algorithm will be able to use a few labeled data points and build a model that can be used for both classification and clustering. The method uses a low dimensional embedding to disclose the hidden micro clusters in the

data. In addition, in order to make the proposed method scalable, two approaches are developed and compared. Finally, the research performed for solving the static semi-supervised algorithm lays the first stone for the development of models for analyzing data streams in an online fashion.

- **Online semi-supervised learning:** The aim is to introduce an online semi-supervised learning algorithm formulated as an optimization problem in the primal and dual setting. We consider the case where new data arrive sequentially but only a small fraction of it is labeled. The available labeled data act as prototypes and help to improve the performance of the algorithm to estimate the labels of the unlabeled data points.

1.5 Overview of Chapters

This thesis is organized in seven chapters. An overview of the chapters is depicted in Fig. 1.2.

Chapter 2: gives a general introduction to kernel functions, Mercer's theorem, and supervised and unsupervised learning using kernel-based methods. In particular an overview of the Least Squares Support Vector Machines for supervised task is provided. In addition the Kernel Spectral Clustering (KSC), a spectral clustering algorithm formulated in the LSSVM optimization framework, is reviewed.

Chapter 3: consists of three main sections. First of all the Least Squares Support Vector machine, which will be used as core model, for regression problems is reviewed. Then the formulation of a method with LSSVM core model is introduced for learning the solution of the given differential equations. In particular, the available initial/boundary conditions are integrated in the learning framework by imposing a set of constraints on the model representing the solution. One of the complications tackled in this chapter is the derivation of the dual kernel based model for expressing the solution of the given differential equations in the dual in terms of the kernel and its derivatives. The presented approach is validated on initial and boundary value problems (IVPs and BVPs) of ordinary differential equations (ODEs) and (DAEs).

Chapter 4: is devoted to parameter estimation of dynamical systems described by ordinary and delay differential equations. A new convex LSSVM based formulation for estimating the unknown parameters of the system within a kernel based framework is presented. The approach consists of two steps. First the trajectories of the differential equation are estimated using the available

observational data. In the second step an optimization problem is formulated for estimating the unknown parameters of the system. Moreover the estimation obtained by the proposed approach is used as an initial guess for solving the original non-convex formulation where the multiple shooting technique is employed. Finally, the proposed method is validated on a number of examples covering constant/time-varying parameter estimation of ordinary and delay differential equations.

Chapter 5: introduces a general framework of non-parallel support vector machines, which involves a regularization term, a scatter loss and a misclassification loss. For binary problems, the framework with proper losses covers some existing non-parallel classifiers. The possibility of incorporating different existing scatter and misclassification loss functions into the framework is investigated. Moreover, a non-parallel semi-supervised algorithm is proposed that can learn from few labeled data points and a large amount of unlabeled data points.

Chapter 6: introduces a novel model called multi-class semi-supervised kernel spectral clustering (MSS-KSC). The model has two modes of implementation: semi-supervised classification and clustering. In this new formulation the labeled data points are incorporated in the objective function of the primal problem through adding a regularization term aiming at minimizing the difference between the latent variables and the labels. Moreover the MSS-KSC algorithm uses a low dimensional embedding to discover the hidden micro clusters. This is highly desirable when the number of existing clusters is large and only few labels from some of them are known a priori. There is also a systematic model selection scheme which is presented as a convex combination of cluster quality index and classification accuracy. The solution vectors are obtained by solving a linear system of equations.

Chapter 7: presents a new algorithm to perform online semi-supervised clustering in a non-stationary environment. The data arrives sequentially and contains only a small number of labeled data points. The available labeled data act as prototypes and help to improve the performance of the algorithm to estimate the labels of the unlabeled data points. Given a few user-labeled data points the initial model is learned and then the class membership of the remaining data points in the current and subsequent time instants are estimated and propagated in an on-line fashion. The update of the memberships is carried out mainly using the out-of-sample extension property of the model. We show how video segmentation can be cast into the online semi-supervised learning framework. In addition we show how to integrates the Kalman filter algorithm into the learning framework by providing an estimation of the labels for the objects in motion in a video sequence.

Chapter 8: Concludes the thesis and proposes future research directions.

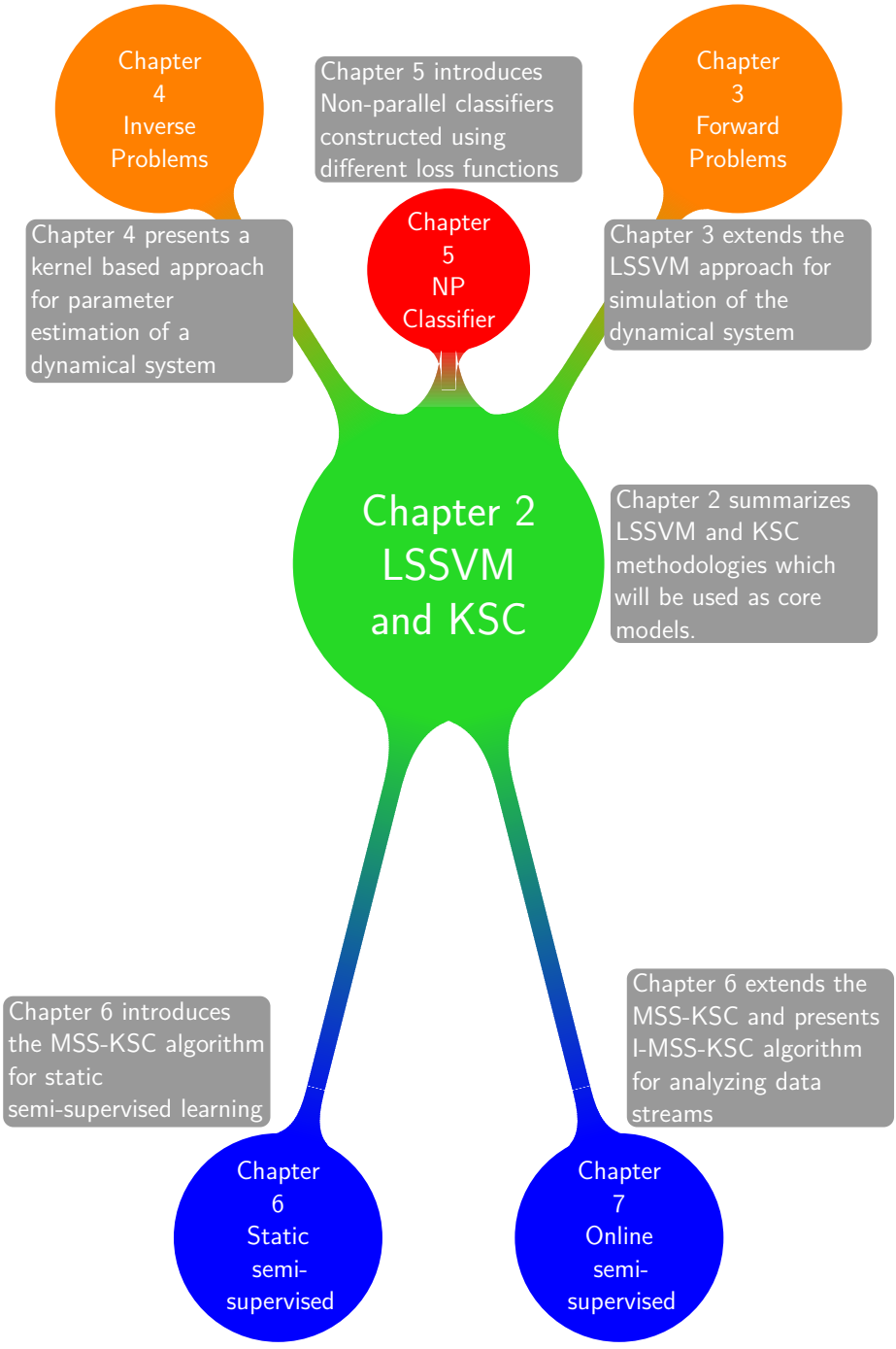


Figure 1.2: The structure of the thesis. Chapters 3, 4, 5, 6 and 7 constitute the main contributions of this thesis.

1.6 Contributions of the Thesis

The main contributions of this work are summarized in the following.

LSSVM based models and learning solutions of dynamical systems:

We propose a methodology based on Least Squares Support Vector machines for simulation of dynamical systems. One starts with representing the solution in the primal in terms of feature maps and then the optimal representation of the solution is obtained in the dual. The solution in the dual is expressed in terms of kernel functions and their derivatives. The initial and boundary conditions are imposed on the primal representation using sets of constraints. The approach is validated on dynamical systems described by ODEs and high index DAEs.

- S. Mehrkanoon, T. Falck, J.A.K. Suykens, "Approximate Solutions to Ordinary Differential Equations Using Least Squares Support Vector Machines", *IEEE Transactions on Neural Networks and Learning Systems,* vol. 23, no. 9, pp. 1356-1367, Sep. 2012.
- S. Mehrkanoon, J.A.K Suykens, "LSSVM approximate solution to linear time varying descriptor systems", *Automatica*, vol. 48, no. 10, pp. 2502-2511, Oct. 2012.
- S. Mehrkanoon, J.A.K Suykens, "Learning Solutions to Partial Differential Equations using LSSVM", *Neurocomputing*, vol 159, pp. 105-116, July. 2015.

LSSVM based models for parameter estimation of dynamical systems:

We present a new algorithm to perform parameter estimation of a given dynamical system. The approach avoids repeated numerical integration of the systems and it uses the ability of the LSSVM for obtaining a closed form solution. Estimation of both constant and time-varying parameters of ordinary and delay differential equations (ODEs and DDEs) are addressed. The approach consists of two steps. In the fist step the trajectory of the given differential equation is approximated by means of LSSVM. The second step includes solving an optimization problem which is constructed based on the information obtained in the first step.

- S. Mehrkanoon, T. Falck, J.A.K. Suykens, "Parameter Estimation for Time Varying Dynamical Systems using Least Squares Support

Vector Machines", *in Proc. of the 16th IFAC Symposium on System Identification (SYSID 2012), Brussels, Belgium*, pp. 1300-1305, Jul. 2012.

- Siamak Mehrkanoon, Saied Mehrkanoon, J.A.K. Suykens, "Parameter estimation of delay differential equations: an integration-free LSSVM approach", *Communication in Nonlinear Science and Numerical Simulation,* vol. 19, no. 4, pp. 830-841, Apr. 2014.

- S. Mehrkanoon, R. Quirynen, M. Diehl, J.A.K. Suykens, "LSSVM based initialization approach for parameter estimation of dynamical systems", *in Proc. of the International Conference on Mathematical Modelling in Physical Sciences (IC-MSQUARE 2013), Prague, Czech Republic,* Sep. 2013.

Nonparallel (semi-)supervised classifiers with different loss functions:

A general framework of non-parallel support vector machines with the possibility of incorporating different combinations of scatter loss and a misclassification loss is introduced. The proposed framework can potentially cover some of the existing non-parallel classifiers. If a certain loss function is used, the method can be viewed as a generalized version of LSSVM core model that is able to produce non-parallel hyperplanes (in case of linear kernel). Furthermore, the approach is extended for tacking problems where few labeled data points and a large amount of unlabeled data points are available.

- S. Mehrkanoon, J.A.K. Suykens, "Non-parallel semi-supervised classification based on kernel spectral clustering", *in Proc. of the International Joint Conference on Neural Networks (IJCNN 2013), Dallas, U.S.A*, pp. 2311-2318, Aug. 2013.

- S. Mehrkanoon, J.A.K. Suykens, "Non-parallel Classifiers with Different Loss Functions", *Neurocomputing*, vol, 143, pp. 294-301, 2014.

Semi-supervised classification and clustering:

A novel multi-class semi-supervised KSC based algorithm called MSS-KSC is introduced to learn from both labeled and unlabeled data points. The problem is formulated as a regularized kernel spectral clustering formulation where the side-information is incorporated to the learning algorithm via a regularization term. The model is obtained by solving a linear system in the dual. Furthermore, the optimal embedding dimension is designed for semi-supervised clustering. This plays a key role when one deals with a large

number of clusters. The proposed method can handle both semi-supervised classification and clustering.

- S. Mehrkanoon, C. Alzate, R. Mall, R. Langone, J.A.K. Suykens, "Multi-class semi-supervised learning based upon kernel spectral clustering", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, pp. 720-733, April 2015.
- S. Mehrkanoon, J.A.K. Suykens, "Large scale semi-supervised learning using KSC based model", *in Proc. of the International Joint Conference on Neural Networks (IJCNN 2014) (IJCNN 2014), Beijing, China*, pp. 4152-4159, Jul. 2014.

Online semi-supervised learning for time evolving data:

A new incremental semi-supervised algorithm called I-MSS-KSC is proposed to analyze data streams that contain few labeled an a lot of unlabeled data points. The approach is the extension of the MSS-KSC towards on-line data clustering, classification. The initially trained model is updated using the out-of-sample extension property of the MSS-KSC model. Moreover for the video segmentation task, the tracking capabilities of the Kalman filter is used to provide the labels of objects in motion and thus regularizing the solution obtained by the MSS-KSC algorithm.

- S. Mehrkanoon, M. Agudelo, J.A.K. Suykens, "Incremental multi-class semi-supervised clustering regularized by Kalman filtering", *Internal Report 14-154, ESAT-SISTA, KU Leuven (Leuven, Belgium)*, 2014, submitted.

Other contributions:

In several occasions the technical expertises acquired in this thesis also contributed to other problems:

- Z. Karevan, S. Mehrkanoon and J.A.K. Suykens, "Black-box modeling for temperature prediction in weather forecasting", *Internal Report 14-154, ESAT-SISTA, KU Leuven (Leuven, Belgium)*, 2015.
- R. Castro, S. Mehrkanoon, A. Marconato, J. Schoukens and J.A.K. Suykens, "SVD truncation schemes for fixed-size kernel models", *in Proc. of the International Joint Conference on Neural Networks (IJCNN), Beijing, China*, pp. 3922-3929, Jun. 2014.

- R. Mall, S. Mehrkanoon, R. Langone, J.A.K. Suykens, “Optimal Reduced Sets for Sparse Kernel Spectral Clustering”, *in Proc. of the International Joint Conference on Neural Networks (IJCNN), Beijing, China*, pp. 2436-2443, Jun. 2014. .

  X. Huang, S. Mehrkanoon, J.A.K. Suykens, "Support Vector Machines with Piecewise Linear Feature Mapping", *Neurocomputing*, vol. 117, pp. 118-127, Oct. 2013.

- R. Mall, S. Mehrkanoon, J.A.K. Suykens, “Identifying Intervals for Hierarchical Clustering using the Gershgorin Circle Theorem”, *Pattern recognition letter*, col 55, pp.1-7, 2015.

Here is also the word cloud of my research projects in the past years:

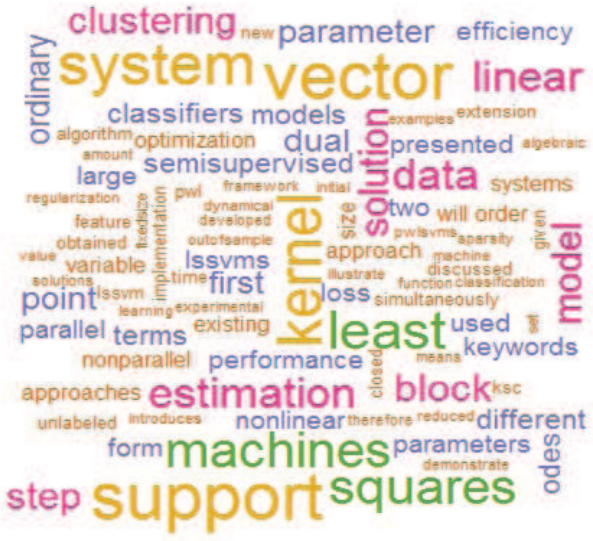


Figure 1.3

Chapter 2

Learning and Kernel Based Models

This chapter reviews the main concepts in kernel-based learning methods for supervised and unsupervised problems. In particular, the primal-dual formulation of the Least Squares Support Vector Machines (LSSVM) for supervised tasks such as classification and regression is discussed. Throughout most of this thesis, the primal-dual formulation of LSSVM based methods plays a central role in the construction of new predictive models used in different domains. Next kernel spectral clustering (KSC), one of the successful unsupervised methods, is reviewed. It enjoys the primal-dual optimization formulation typical of LSSVM based models. The main advantages of kernel spectral clustering over the classical spectral clustering is the existence of a model selection scheme and the out-of-sample extension property to unseen data. In Chapters 6 and 7, the KSC method will be used as core model for the construction of the semi-supervised clustering/classification algorithms.

2.1 Kernel Methods

The work in this thesis is developed using the Least Squares Support Vector Machines (LSSVM) [156] formulation as a core model. Support Vector Machines (SVMs) [167] and Least Squares SVM follow the approach of a primal-dual optimization formulation, where both techniques make use of a so-called feature space where the inputs have been transformed by means of a nonlinear

mapping. This is converted to the dual space by means of Mercer's theorem and the use of a positive definite kernel, without computing explicitly the mapping.

Other directions in kernel methods follow different approaches. For instance in Reproducing Kernel Hilbert Spaces (RKHS) [50] the problem of function estimation is treated as a variational problem and Gaussian Processes (GP) [139] follow a probabilistic Bayesian setting. Although these different approaches have links with each other, in general the methodologies are different. In particular, the primal-dual formulation of LSSVM makes it easy to add additional constraints, and therefore makes it straightforward to integrate more prior knowledge into the models.

2.2 Least Squares Support Vector Machines

The Support Vector Machine (SVM) is a powerful methodology for solving pattern recognition and function estimation problems. In this method one maps the data into a high dimensional feature space and performs linear classification, which corresponds to a non-linear decision boundary in the original input space. The dual solution of SVM formulation is a quadratic programming problem. On the other hand, LSSVMs for function estimation, classification, problems in unsupervised learning and others has been investigated in [156]. In this case, the problem formulation involves equality instead of inequality constraints. This leads to a system of linear equations at the dual level, in the context of regression and classification.

2.2.1 Regression Problem

Consider a given training set $\{x_i, y_i\}_{i=1}^n$ with input data $x_i \in \mathbb{R}^d$ and output data $y_i \in \mathbb{R}$. The goal is to estimate a model of the form

$$\hat{y}(x) = w^T \varphi(x) + b.$$

The primal LSSVM model for regression can be written as follows [156]

$$\begin{aligned} \underset{\boldsymbol{w},b,\boldsymbol{e}}{\text{minimize}} \quad & \frac{1}{2} w^T w + \frac{\gamma}{2} e^T e \\ \text{subject to} \quad & y_i = w^T \varphi(x_i) + b + e_i, \quad i = 1, ..., n, \end{aligned} \tag{2.1}$$

where $\gamma \in \mathbb{R}^+, b \in \mathbb{R}$, $w \in \mathbb{R}^h$. $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^h$ is the feature map and h is the dimension of the feature space. The dual solution is then given by

$$\left[\begin{array}{c|c} \Omega + I_n/\gamma & 1_n \\ \hline 1_n^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline b \end{array}\right] = \left[\begin{array}{c} y \\ \hline 0 \end{array}\right]$$

where $\Omega_{ij} = K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$ is the ij-th entry of the positive definite kernel matrix. $1_n = [1, \ldots, 1]^T \in \mathbb{R}^n$, $\alpha = [\alpha_1, \ldots, \alpha_n]^T$, $y = [y_1, \ldots, y_n]^T$ and I_n is the identity matrix. The model in the dual form becomes:

$$\hat{y}(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b.$$

It should be noted that if $b = 0$, for an explicitly known and finite dimensional feature map φ the problem could be solved in the primal (ridge regression if b=0) by eliminating e and then w would be the only unknown. But in the LSSVM approach the feature map φ is not explicitly known in general and can be infinite dimensional. Therefore the kernel trick is used and the problem is solved in the dual.

In the subsequent chapters, the constrained optimization framework with the LSSVM as a code model, will be used in the context of learning the solution of dynamical system and unknown constant/time-varying parameter estimation of parameter affine dynamical systems.

2.2.2 Classification Problem

Given a training data set $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ are the training points and $y_i \in \{-1, 1\}$ are the class labels, the convex primal problem of the LSSVM classifier can be formulated as [157, 156]:

$$\begin{aligned} \min_{w, e_i, b} \quad & \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^{n} e_i^2 \\ \text{subject to} \quad & y_i(w^T \varphi(x_i) + b) = 1 - e_i, i = 1, \ldots, n. \end{aligned} \tag{2.2}$$

The model in the primal space is expressed in terms of the feature map i.e. $\hat{y} = w^T \varphi(x) + b$. The e_i are slack variables allowing deviations from the target value 1. The regularization parameter γ controls the trade-off between the regularization term and the minimization of the training error. The Lagrangian

of (2.2) takes the following form:

$$\mathcal{L}(w, e_i, b, \alpha_i) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^{n} e_i^2 - \sum_{i=1}^{n} \alpha_i (y_i (w^T \varphi(x_i) + b) - 1 + e_i) \quad (2.3)$$

where α_i are the Lagrange multipliers. The KKT optimality conditions are:

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i), \\
&\frac{\partial \mathcal{L}}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i, \\
&\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0, \\
&\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow y_i (w^T \varphi(x_i) + b) - 1 + e_i = 0.
\end{aligned}$$

Eliminating the primal variables e_i and w leads to the following linear system in the dual problem:

$$\left[\begin{array}{c|c} \tilde{\Omega} + I_n/\gamma & y \\ \hline y^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline b \end{array}\right] = \left[\begin{array}{c} 1_n \\ \hline 0 \end{array}\right] \quad (2.4)$$

where $y = [y_1, \ldots, y_n]^T$, $1_n = [1, \ldots, 1]^T$, $\alpha = [\alpha_1, \ldots, \alpha_n]^T$. The kernel matrix is denoted by $\tilde{\Omega}$ with entries $\tilde{\Omega}_{ij} = y_i y_j \varphi(x_i)^T \varphi(x_j) = y_i y_j K(x_i, x_j)$, where $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is the kernel function which maps the input data points into the high dimensional feature space $\varphi(\cdot)$. The LSSVM classification model in the dual becomes:

$$\hat{y}(x) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i K(x, x_i) + b). \quad (2.5)$$

2.3 Kernel Spectral Clustering

Unsupervised learning techniques like principal component analysis (PCA) and clustering aim at finding the underlying complex structure of a given unlabeled

data points. In clustering one seeks to find partitions (clusters) that consist of objects that are similar to each other and dissimilar to objects in other clusters. Some of the well-known clustering algorithms are for instance k-means and spectral clustering. In this section a more recent and advanced clustering algorithm, kernel spectral clustering (KSC), which was originally proposed in [4] and will be described.

Kernel Spectral Clustering (KSC) corresponds to a weighted kernel PCA formulation and represents a spectral clustering formulation in the LSSVM optimization framework with primal and dual representations. The solution in the dual is obtained by solving an eigenvalue problem, related to spectral clustering [4]. However, as opposed to classical spectral clustering, the KSC method possesses a natural extension to out-of-sample data i.e. the possibility to apply the trained clustering model to unseen data points. In addition it can enjoy a good generalization performance due the existence of the model selection scheme. The model can be trained using the training data points (a subset of the full data) and then be used to predict the membership of the unseen test data points in a learning framework.

Given training data $\mathcal{D} = \{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^d$, the primal problem of kernel spectral clustering is formulated as follows [4]:

$$\begin{aligned} \min_{w^{(\ell)},b^{(\ell)},e^{(\ell)}} \quad & \frac{1}{2}\sum_{\ell=1}^{N_c-1} w^{(\ell)^T} w^{(\ell)} - \frac{1}{2n}\sum_{\ell=1}^{N_c-1} \gamma_\ell e^{(\ell)^T} V e^{(\ell)} \\ \text{subject to} \quad & e^{(\ell)} = \Phi w^{(\ell)} + b^{(\ell)} 1_n,\ \ell = 1,\ldots,N_c-1 \end{aligned} \tag{2.6}$$

where N_c is the number of desired clusters, $e^{(\ell)} = [e_1^{(\ell)},\ldots,e_n^{(\ell)}]^T$ are the projected variables (score variables) and $\ell = 1,\ldots,N_c-1$ indicates the number of score variables required to encode the N_c clusters. $\gamma_\ell \in \mathbb{R}^+$ are the regularization constants. Here

$$\Phi = [\varphi(x_1),\ldots,\varphi(x_n)]^T \in \mathbb{R}^{n\times h}$$

where $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^h$ is the feature map and h is the dimension of the feature space which can be infinite dimensional. A vector of all ones with size n is denoted by 1_n. $w^{(\ell)}$ is the model parameters vector in the primal. $V = \text{diag}(v_1,...,v_n)$ with $v_i \in \mathbb{R}^+$ is a user defined weighting matrix.

Applying the Karush-Kuhn-Tucker (KKT) optimality conditions one can show that the solution in the dual can be obtained by solving an eigenvalue problem of the following form:

$$V P_v \Omega \alpha^{(\ell)} = \lambda \alpha^{(\ell)}, \tag{2.7}$$

where $\lambda = n/\gamma_\ell$, $\alpha^{(\ell)}$ are the Lagrange multipliers and P_v is the weighted centering matrix:

$$P_v = I_n - \frac{1}{1_n^T V 1_n} 1_n 1_n^T V,$$

where I_n is the $n \times n$ identity matrix and Ω is the kernel matrix with ij-th entry $\Omega_{ij} = K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$. The effect of P_v is to center the kernel matrix Ω by removing the weighted mean from each column. As a result, the eigenvectors are zero-mean. Given this and due to the fact that the eigenvectors are piecewise constant, it is possible to use the eigenvectors corresponding to the first $N_c - 1$ eigenvalues to partition the dataset into N_c clusters. In the ideal case of N_c well separated clusters, for a properly chosen kernel parameter, the matrix $VP_v\Omega$ has $N_c - 1$ piecewise constant eigenvectors with eigenvalue 1.

The eigenvalue problem (2.7) is related to spectral clustering with random walk Laplacian. In this case, the clustering problem can be interpreted as finding a partition of the graph in such a way that the random walker remains most of the time in the same cluster with few jumps to other clusters, minimizing the probability of transitions between clusters. It is shown that if

$$V = D^{-1} = \text{diag}(\frac{1}{d_1}, \cdots, \frac{1}{d_n}),$$

where $d_i = \sum_{j=1}^n K(x_i, x_j)$ is the degree of the i-th data point, the dual problem is related to the random walk algorithm for spectral clustering.

From the KKT optimality conditions one can show that the score variables $e^{(\ell)}$ can be written as follows:

$$e^{(\ell)} = \Phi w^{(\ell)} + b^{(\ell)} 1_n = \Phi\Phi^T \alpha^{(\ell)} + b^{(\ell)} 1_n$$

$$= \Omega\alpha^{(\ell)} + b^{(\ell)} 1_n,\ \ell = 1, \ldots, N_c - 1.$$

The out-of-sample extensions to test points $\{x_i\}_{i=1}^{n_{\text{test}}}$ is done by an Error-Correcting Output Coding (ECOC) decoding scheme. First the cluster indicators are obtained by binarizing the score variables for test data points as follows:

$$q_{\text{test}}^{(\ell)} = \text{sign}(e_{\text{test}}^{(\ell)}) = \text{sign}(\Phi_{\text{test}} w^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}})$$

$$= \text{sign}(\Omega_{\text{test}} \alpha^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}}),$$

where $\Phi_{\text{test}} = [\varphi(x_1), \ldots, \varphi(x_{n_{\text{test}}})]^T$ and $\Omega_{\text{test}} = \Phi_{\text{test}} \Phi^T$. The decoding scheme consists of comparing the cluster indicators obtained in the test stage

Algorithm 1: KSC algorithm [4]

Data: Training set $\mathcal{D} = \{x_i\}_{i=1}^{n}$, test set $\mathcal{D}^{test} = \{x_i^{test}\}_{i=1}^{N_{\text{test}}}$, kernel parameters (if any), number of clusters N_c.

Result: Clusters $\{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$, codebook $\mathcal{CB} = \{c_q\}_{q=1}^{N_c}$ with $\{c_q\} \in \{-1, 1\}^{N_c-1}$.

1 compute the training eigenvectors $\alpha^{(l)}$, $l = 1, \ldots, N_c - 1$, corresponding to the $N_c - 1$ largest eigenvalues of problem (2.7)

2 Binarize the eigenvectors matrix $A = [\alpha^{(1)}, \ldots, \alpha^{(N_c-1)}]$ and form the code-book $\mathcal{CB} = \{c_q\}_{q=1}^{k}$ using the N_c most occurrences encodings of $\text{sign}(A)$.

3 $\forall i$, $i = 1, \ldots, n$, assign x_i to A_{q^*} where $q^* = \underset{q}{\text{argmin}} d_H(\text{sign}(\alpha_i), c_q)$, where $d_H(.,.)$ is the Hamming distance

4 Compute the cluster indicators for test data $\text{sign}(e_m^{(l)})$, $m = 1, \ldots, N_{\text{test}}$, and let $\text{sign}(e_m) \in \{-1, 1\}^{N_c-1}$ be the encoding vector of x_m^{test}

5 $\forall m$, assign x_m^{test} to A_{q^*}, where $q^* = \underset{q}{\text{argmin}} d_H(\text{sign}(e_m), c_q)$.

with the codebook (which is obtained in the training stage) and selecting the nearest codeword in terms of Hamming distance.

The KSC method is summarized in Algorithm. 1. KSC is provided with a model selection scheme based on the Balanced Line Fit (BLF) criterion [4]. It can be shown that in the ideal situation of well separated clusters, the data projections (score variables e_i) associated with the KSC formulation, form lines one per each cluster. The shape of the data points in the projections space, is exploited by the BLF criterion to select the optimal clustering parameters e.g. the number of clusters (k) and the kernel bandwidth σ. The BLF criterion is defined as follows [4]:

$$\text{BLF}(\mathcal{D}^{\text{Val}}, N_c) = \eta \text{linefit}(\mathcal{D}^{\text{Val}}, N_c) + (1 - \eta)\text{balance}(\mathcal{D}^{\text{Val}}, N_c) \tag{2.8}$$

where $\mathcal{D}^{\text{Val}}$ represents the validation set and N_c indicates the number of clusters. The linefit index equals 0 when the score variables are distributed spherically and equals 1 when the score variables are collinear, representing points in the same cluster. The balance index equals 1 when the clusters have the same number of elements and tends to 0 in extremely unbalanced cases. The parameter η controls the importance given to the linefit with respect to the balance index and takes values in the range $[0, 1]$.

Later, In Chapter 6 and 7, the KSC model will serve as a core model in the development of multi-class semi-supervised learning algorithm.

Chapter 3

Learning Solutions of Dynamical Systems

In this chapter, kernel based approaches are formulated to learn the solution of different types of differential equations including Ordinary Differential Equations (ODEs), Differential Algebraic Equations (DAEs) and Partial Differential Equations (PDEs). The optimal representation of the solution is obtained in the primal-dual setting. The model is built by incorporating the initial/boundary conditions as constraints of an optimization problem. The approximate solution is presented in closed form by means of LSSVMs, whose parameters are adjusted to minimize an appropriate error function. For the linear and nonlinear cases, these parameters are obtained by solving a system of linear and nonlinear equations respectively.

3.1 Related Work

Differential equations can be found in the mathematical formulation of physical phenomena in a wide variety of applications especially in science and engineering [45, 88]. This chapter focuses on three types of differential equations such as ordinary/partial differential equations as well as differential algebraic equations. In contrast to ordinary differential equations (ODEs), which deal with functions of a single independent variable i.e. time and their derivatives, partial differential equations (PDEs) are used to formulate problems involving functions of several independent variables. In other

words, ODEs model one-dimensional dynamical systems and PDEs model multidimensional systems.

- ODEs: Depending upon the form of the boundary conditions to be satisfied by the solution, problems involving ODEs can be divided into two main categories, namely initial value problems (IVPs) and boundary value problems (BVPs). Analytic solutions for these problems are not generally available and hence numerical methods must be applied.

  Many methods have been developed for solving initial value problems of ODEs, such as Runge-Kutta, finite difference, predictor-corrector and collocation methods [35, 90, 43, 140]. Generally speaking numerical methods for approximating the solution of the boundary value problems fall into two classes: the difference methods and shooting methods. In the shooting method, one tries to reduce the problem to initial value problems by providing a sufficiently good approximation of the derivative values at the initial point.

- DAEs: Differential algebraic equations (DAEs) arise frequently in numerous applications including mathematical modelling, circuit and control theory [33], chemistry [60, 138], fluid dynamic [100] and computer-aided design. DAEs have been known under a variety of names, depending on the area of application for instance they are also called descriptor, implicit or singular systems. The most general form of DAE is given by

  $$F(\dot{x}, x, t) = 0 \tag{3.1}$$

  where $\frac{\partial F}{\partial \dot{x}}$ is singular. The rank and structure of this Jacobian matrix depends, in general, on the solution $x(t)$. DAEs are characterized by their index. In [34] the index of (3.1) is defined as the minimum number of differentiations of the system which would be required to solve for $\dot{x}$ uniquely in terms of x and t. The index of DAEs is a measure of the degree of singularity of the system and it is widely considered as an indication of certain difficulties for numerical methods. We note that DAEs with an index greater than 1 are often referred to as higher-index DAEs and that the index of an ODE is zero. See [11, 36] for a detailed discussion of the index of a DAE. The important special case of (3.1) is semi-explicit DAE or an ODE with constraints i.e.

  $$\begin{aligned} \dot{x} &= f(x, y, t) \\ 0 &= g(x, y, t). \end{aligned} \tag{3.2}$$

  The index is 1 if $\frac{\partial g}{\partial y}$ is nonsingular. x and y are considered as differential and algebraic variables respectively. Analytic solutions for these problems

are not generally available and hence numerical methods must be applied. Some numerical methods have been developed for solving DAEs using backward differentiation formulas (BDF) [9, 34, 63, 11, 12] or implicit Runge-Kutta (IRK) methods [34, 12, 10, 124]. These methods are only applicable to low-index problems and often require the problem to have special structure. Furthermore, these approaches approximate the solutions at discrete points only (discrete solution) and some interpolation technique is needed in order to get a continuous solution. Thereby, recently attempts have been made to develop methods that produce a closed form approximate solution. Awawdeh et al. [14] applied Homotopy analysis method to systems of DAEs. The authors in [68] used Padé approximation methods to estimate the solution of singular systems with index-2.

In general, the higher the index, the greater numerical difficulty one is going to encounter when solving differential algebraic equations numerically and an alternative treatment is the use of index reduction techniques which are based on repeated differentiation of the constraints until a low-index problem (an index 1 DAE or ODE) is obtained. There are several reasons to consider differential algebraic equations (3.2) directly, rather than convert them to system of ODEs [34, 172]. Therefore designing direct methods that do not require a reformulation (e.g. index reduction) of DAEs will not only speed up the solution, also the system structure (e.g. the modelling changes and parameter variations) can be more readily explored.

It is known that the singular system can have instantaneous jumps due to inconsistent initial conditions. Several approaches for consistent initialization of DAEs have been studied in the literature and in general they fall into two categories: rigorous initialization and the direct initialization method (we refer the reader to [96] and references therein for more details). Within the scope of this thesis we assume that consistent initial conditions are available.

- PDEs: In most applications the analytic solutions of the underlying PDEs are not available and therefore numerical methods must be applied. For that reason, a number of numerical methods such as Finite Difference methods (FDM) [89, 53, 79, 151, 75, 120], Finite Element methods (FEM) [161, 169, 95], Splines [1, 8], Multigrid methods [74, 166, 69] and methods based on neural networks [47, 137, 106, 163, 86, 146] and genetic programming approaches [149, 164, 129] have been developed.

  The finite difference methods provide the solution at specific preassigned mesh points only (discrete solution) and they need an additional interpolation procedure to yield the solution for the whole domain. The

finite-element method (FEM) is the most popular discretization method in engineering applications. An important feature of the FEM is that it requires a discretization of the domain via meshing, and therefore belongs to the class of mesh-based methods.

Another class of methods that can generate a closed form solution and do not require meshing are based on neural network models see [107, 93, 165, 163]. Lee and Kang [93] used neural networks models to solve first order differential equations. They do not require a mesh topology and the domain of interest is presented by scattered discrete points. The authors in [86] introduced a method based on feedforward neural networks to solve ordinary and partial differential equations. In that model, the approximate solution was chosen such that it, by construction, satisfied the supplementary conditions. Therefore the model function was expressed as a sum of two terms. The first term, which contains no adjustable parameters, satisfied the initial/boundary conditions and the second term involved a feedforward neural network to be trained.

Despite the fact that the classical neural networks have nice properties such as universal approximation, they still suffer from having two persistent drawbacks. The first problem is the existence of many local minima solutions. The second problem is how to choose the number of hidden units.

Support Vector Machines (SVMs) are a powerful methodology for solving pattern recognition and function estimation problems [142, 167]. In this method one maps data into a high dimensional feature space and there solves a linear regression problem. It leads to solving quadratic programming problems. LSSVMs for function estimation, classification, problems in unsupervised learning and others has been investigated in [157, 158, 52, 130]. In this case, the problem formulation involves equality instead of inequality constraints. The training for regression and classification problems is then done by solving a set of linear equations.

We propose a kernel based method in the LSSVM framework for learning the solution of a dynamical system. It should be noted that one can derive a kernel based model in two ways: one is using a primal-dual setting and the other one is by using function estimation in a reproducing kernel Hilbert space and the corresponding representer theorem. The primal-dual approach has the advantage that it is usually straightforward to incorporate additional structure or primal knowledge into the estimation problem. For instance in the context of learning the solution of PDEs, one may know in advance that the underlying solution has to satisfy an additional constraint (like non-local conservation condition [6]). Then one can incorporate it to the estimation problem by adding a suitable set of constraints. Furthermore, the primal and dual formulation of the method allows to obtain the optimal representation of the solution. That

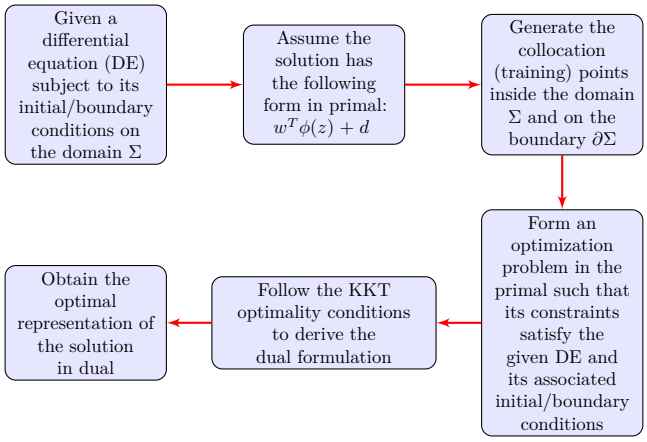


Figure 3.1: The general steps of the process from the representation of the solution in the primal to dual.

means in the primal one starts with a simple representation of the solution and by incorporating the initial/boundary conditions together with the system dynamics, one may obtain the optimal representation of the solution in the dual. That is in contrast with most existing approaches that produce a closed form solution. More precisely, unlike the approach described in [87] that the user has to define a form of a trial solution, which in some cases is not straightforward, in the proposed approach the optimal model is derived by incorporating the initial/boundary conditions as constraints of an optimization problem. The interaction between three main counterparts playing in this chapter is shown in Fig. 3.2. The general stages (methodology) of the procedure are described by the flow-chart 3.1.

3.2 Learning the Solution of ODEs

This section describes the problem statement. After that the operators that will be used in the subsequent sections are defined.

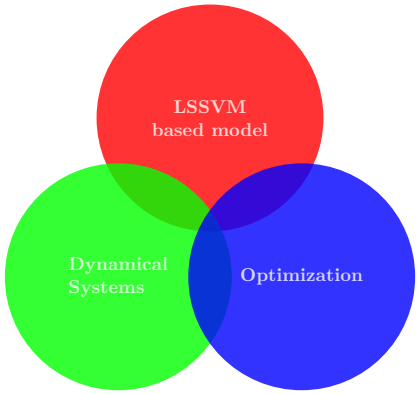


Figure 3.2: Interaction between three main counterparts

3.2.1 Problem statement and overview of existing methods

Consider the general m-th order linear ordinary differential equation with time varying coefficients of the form

$$\mathcal{L}[y] \equiv \sum_{\ell=0}^{m} f_\ell(t) y^{(\ell)}(t) = r(t), \quad t \in [a, c] \tag{3.3}$$

where $\mathcal{L}$ represents an m-th order linear differential operator, $[a, c] \subset \mathbb{R}$ is the problem domain and $r(t)$ is the input signal. $f_\ell(t)$ are known functions and $y^{(\ell)}(t)$ denotes the ℓ-th derivative of y with respect to t. The $m-1$ necessary initial or boundary conditions for solving the above differential equations are:
IVP:

$$\mathcal{IC}_\mu[y(t)] = p_\mu, \quad \mu = 0, ..., m-1;$$

BVP:

$$\mathcal{BC}_\mu[y(t)] = q_\mu, \quad \mu = 0, ..., m-1,$$

where $\mathcal{IC}_\mu$ are the initial conditions (all constraints are applied at the same value of the independent variable i.e. $t = a$) and $\mathcal{BC}_\mu$ are the boundary conditions (the constraints are applied at multiple values of the independent variable t, typically at the ends of the interval $[a, c]$ in which the solution is sought). p_μ and q_μ are given scalars.

A differential equation (3.3) is said to be stiff when its exact solution consists of a steady state term that does not grow significantly with time, together with a transient term that decays exponentially to zero. Problems involving rapidly decaying transient solutions occur naturally in a wide variety of applications,

including the study of damped mass spring systems and the analysis of control systems (see [90] for more details).

The approaches given in [86], define a trial solution to be a sum of two terms i.e. $y(t) = H(t) + F(t, N(t, P))$. The first term $H(t)$, which has to be defined by the user and in some cases is not straightforward, satisfies the initial/boundary conditions and the second term $F(t, N(t, P))$ is a single-output feed forward neural network with input t and parameters P. In contrast with the approaches given in [86], we build the model by incorporating the initial/boundary conditions as constraints of an optimization problem. Therefore the task of defining a trial solution which can potentially be a difficult problem is avoided and instead the optimal representation of the solution is learned using an optimization framework.

3.2.2 Definitions of some operators

Let us assume an explicit model $\hat{y}(t) = w^T\varphi(t) + b$ as an approximation for the solution of the differential equation. Since there are no data available in order to learn from the differential equation, we have to substitute our model into the given differential equation. Therefore we need to define the derivative of the kernel function. Making use of Mercer's Theorem [167], derivatives of the feature map can be written in terms of derivatives of the kernel function [91]. Let us define the following differential operator which will be used in subsequent sections

$$\nabla_n^m \equiv \frac{\partial^{n+m}}{\partial u^n \partial v^m}.$$

If $\varphi(u)^T\varphi(v) = K(u, v)$, then one can show that

$$[\varphi^{(n)}(u)]^T\varphi^{(m)}(v) = \nabla_n^m[\varphi(u)^T\varphi(v)] = \nabla_n^m[K(u, v)] = \frac{\partial^{n+m}K(u, v)}{\partial u^n \partial v^m}. \quad (3.4)$$

Using formula (3.4), it is possible to express all derivatives of the feature map in terms of the kernel function itself (provided that the kernel function is sufficiently differentiable). For instance the following relations hold,

$$\nabla_1^0[K(u, v)] = \frac{\partial(\varphi(u)^T\varphi(v))}{\partial u} = \varphi^{(1)}(u)^T\varphi(v),$$

$$\nabla_0^1[K(u, v)] = \frac{\partial(\varphi(u)^T\varphi(v))}{\partial v} = \varphi(u)^T\varphi^{(1)}(v),$$

$$\nabla_2^0[K(u, v)] = \frac{\partial^2(\varphi(u)^T\varphi(v))}{\partial u^2} = \varphi^{(2)}(u)^T\varphi(v).$$

One starts with a general approximate solution to (3.3) of the form of $\hat{y}(t) = w^T\varphi(t) + b$, where w and b are unknowns of the model that have to be determined. To obtain the optimal value of these parameters, collocation methods can be used which assume a discretization of the interval $[a, c]$ into a set of collocation points $\Upsilon = \{a = t_1 < t_2 < \cdots < t_N = c\}$. Therefore the w and b are to be found by solving the following optimization problem:

For the IVP Case:

$$\begin{aligned} \underset{\hat{y}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N}\Big[(\mathcal{L}[\hat{y}] - r)(t_i)\Big]^2 \\ \text{subject to} \quad & \mathcal{IC}_\mu[\hat{y}(t)] = p_\mu, \quad \mu = 0, ..., m-1. \end{aligned} \tag{3.5}$$

For the BVP case:

$$\begin{aligned} \underset{\hat{y}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N}\Big[(\mathcal{L}[\hat{y}] - r)(t_i)\Big]^2 \\ \text{subject to} \quad & \mathcal{BC}_\mu[\hat{y}(t)] = q_\mu, \quad \mu = 0, ..., m-1, \end{aligned} \tag{3.6}$$

where N is the number of collocation points (which is equal to the number of training points) used to undertake the learning process. In what follows we formulate the optimization problem in the LSSVM framework for solving linear ordinary differential equations. For notational convenience let us list the following notations which are used in the following sections,

$$[\nabla_n^m K](t, s) = [\nabla_n^m K(u, v)]\Big|_{u=t, v=s},$$

$$\Omega_n^m(i, j) = \nabla_n^m[K(u, v)]\Big|_{u=t_i, v=t_j} = \frac{\partial^{n+m}K(u, v)}{\partial u^n \partial v^m}\Big|_{u=t_i, v=t_j},$$

$$\Omega_0^0(i, j) = \nabla_0^0[K(u, v)]\Big|_{u=t_i, v=t_j} = K(t_i, t_j).$$

Here $\Omega_n^m(i, j)$ denotes the (i, j)-th entry of matrix Ω_n^m. The notation $M(k : l, m : n)$ is used for selecting a submatrix of matrix M consisting of rows k to l and columns m to n. $M(i, :)$ denotes the i-th row of matrix M. $M(:, j)$ denotes the j-th column of matrix M.

3.2.3 Formulation of the method for first order IVP

As a first example consider the following first order initial value problem,

$$y'(t) - f_1(t)y(t) = r(t), \quad y(a) = p_1, \quad a \leq t \leq c. \tag{3.7}$$

In the LSSVM framework the approximate solution can be obtained by solving the following optimization problem,

$$\begin{aligned} \underset{w,b,e}{\text{minimize}} \quad & \frac{1}{2}w^T w + \frac{\gamma}{2}e^T e \\ \text{subject to} \quad & w^T\varphi'(t_i) = f_1(t_i)\Big[w^T\varphi(t_i) + b\Big] + \\ & r(t_i) + e_i, \quad i = 2, ..., N, \\ & w^T\varphi(t_1) + b = p_1. \end{aligned} \tag{3.8}$$

This problem is obtained by combining the LSSVM cost function with constraints constructed by imposing the approximate solution $\hat{y}(t) = w^T\varphi(t) + b$, given by the LSSVM model, to satisfy the given differential equation with corresponding initial condition at collocation points $\{t_i\}_{i=1}^N$. Problem (3.8) is a quadratic minimization under linear equality constraints, which enables an efficient solution.

Lemma 3.2.1. *Given a positive definite kernel function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with $K(t, s) = \varphi(t)^T\varphi(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (3.8) is obtained by solving the following dual problem [110]:*

$$\left[\begin{array}{c|c|c} \mathcal{K} + I_{N-1}/\gamma & h_{p_1} & -f_1 \\ \hline h_{p_1}^T & 1 & 1 \\ \hline -f_1^T & 1 & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \\ \hline b \end{array}\right] = \left[\begin{array}{c} r \\ \hline p_1 \\ \hline 0 \end{array}\right] \tag{3.9}$$

with

$$\alpha = [\alpha_2, \ldots, \alpha_N]^T, f_1 = [f_1(t_2), \ldots, f_1(t_N)]^T \in \mathbb{R}^{N-1},$$

$$r = [r(t_2), \ldots, r(t_N)]^T \in \mathbb{R}^{N-1},$$

$$\mathcal{K} = \tilde{\Omega}_1^1 - D_1\tilde{\Omega}_1^0 - \tilde{\Omega}_0^1 D_1 + D_1\tilde{\Omega}_0^0 D_1,$$

$$h_{p_1} = [\Omega_0^1(1, 2:N)]^T - D_1[\Omega_0^0(1, 2:N)]^T.$$

D_1 is a diagonal matrix with the elements of f_1 on the main diagonal. $\tilde{\Omega}_n^m = [\Omega_n^m]_{2:N,2:N}$ for $n, m = 0, 1$. Also note that $\mathcal{K} \in \mathbb{R}^{(N-1)\times(N-1)}$ and $h_{p_1} \in \mathbb{R}^{N-1}$.

Proof. The Lagrangian of the constrained optimization problem (3.8) becomes

$$\mathcal{L}(w, b, e_i, \alpha_i, \beta) =$$

$$\frac{1}{2} w^T w + \frac{\gamma}{2} e^T e - \sum_{i=2}^{N} \alpha_i \Big[w^T \Big(\varphi'(t_i) - f_1(t_i)\varphi(t_i) \Big)$$

$$- f_1(t_i) b - r_i - e_i \Big] - \beta \Big(w^T \varphi(t_1) + b - p_1 \Big)$$

where $\{\alpha_i\}_{i=2}^N$ and β are Lagrange multipliers and $r_i = r(t_i)$ for $i = 2, ..., N$. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \to w = \sum_{i=2}^{N} \alpha_i \Big(\varphi'(t_i) - f_1(t_i)\varphi(t_i) \Big) + \beta\varphi(t_1),$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \to \sum_{i=2}^{N} \alpha_i f_1(t_i) - \beta = 0,$$

$$\frac{\partial \mathcal{L}}{\partial e_i} = 0 \to e_i = -\frac{\alpha_i}{\gamma}, \quad i = 2, ..., N,$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \to w^T \Big(\varphi'(t_i) - f_1(t_i)\varphi(t_i) \Big) - f_1(t_i) b - e_i = r_i, \quad i = 2, ..., N,$$

$$\frac{\partial \mathcal{L}}{\partial \beta} = 0 \to w^T \varphi(t_1) + b = p_1.$$

After elimination of the primal variables w and $\{e_i\}_{i=2}^N$ and making use of Mercer's Theorem, the solution is given in the dual by

$$\begin{cases} r_i = \sum_{j=2}^{N} \alpha_j \Big[\Omega_1^1(j,i) - f_1(t_i)\Big(\Omega_1^0(j,i) - f_1(t_j)\Omega_0^0(j,i) \Big) \\ \quad -f_1(t_j)\Omega_0^1(j,i) \Big] + \beta \Big(\Omega_0^1(1,i) - f_1(t_i)\Omega_0^0(1,i) \Big) \\ \quad + \frac{\alpha_i}{\gamma} - f_1(t_i)b, \quad i = 2, ..., N, \\ p_1 = \sum_{j=2}^{N} \alpha_j \Big(\Omega_1^0(j,1) - f_1(t_j)\Omega_0^0(j,1) \Big) + \beta\, \Omega_0^0(1,1) + b, \\ 0 = \sum_{j=2}^{N} \alpha_j f_1(t_j) - \beta \end{cases}$$

and writing these equations in matrix form gives the linear system in (5.11). □

The model in the dual form becomes

$$\hat{y}(t) = \sum_{i=2}^{N} \alpha_i \Big([\nabla_1^0 K](t_i, t) - f_1(t_i)[\nabla_0^0 K](t_i, t) \Big) + \beta\, [\nabla_0^0 K](t_1, t) + b$$

where K is the kernel function.

3.2.4 Formulation of the method for second order IVP and BVP

IVP case:
Let us consider a second order IVP of the form,

$$y''(t) = f_1(t)y'(t) + f_2(t)y(t) + r(t), \quad t \in [a, c]$$

$$y(a) = p_1, \quad y'(a) = p_2.$$

The approximate solution, $\hat{y}(t) = w^T\varphi(t) + b$, is then obtained by solving the following optimization problem,

$$\begin{aligned} \underset{w,b,e}{\text{minimize}} \quad & \frac{1}{2}w^T w + \frac{\gamma}{2}e^T e \\ \text{subject to} \quad & w^T\varphi''(t_i) = f_1(t_i)w^T\varphi'(t_i) + \\ & f_2(t_i)[w^T\varphi(t_i) + b] + r(t_i) + e_i, \quad i = 2, ..., N, \\ & w^T\varphi(t_1) + b = p_1, \\ & w^T\varphi'(t_1) = p_2. \end{aligned} \tag{3.10}$$

Lemma 3.2.2. *Given a positive definite kernel function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with $K(t, s) = \varphi(t)^T\varphi(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (3.10) is obtained by solving the following dual problem [110]:*

$$\left[\begin{array}{c|c|c|c} \mathcal{K} + I_{N-1}/\gamma & h_{p_1} & h_{p_2} & -f_2 \\ \hline h_{p_1}^T & 1 & 0 & 1 \\ \hline h_{p_2}^T & 0 & \Omega_1^1(1,1) & 0 \\ \hline -f_2^T & 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta_1 \\ \hline \beta_2 \\ \hline b \end{array}\right] = \left[\begin{array}{c} r \\ \hline p_1 \\ \hline p_2 \\ \hline 0 \end{array}\right] \tag{3.11}$$

where

$$\begin{aligned} & \alpha = [\alpha_2, \ldots, \alpha_N]^T, \ f_1 = [f_1(t_2), \ldots, f_1(t_N)]^T \in \mathbb{R}^{N-1}, \\ & f_2 = [f_2(t_2), \ldots, f_2(t_N)]^T \in \mathbb{R}^{N-1}, \\ & r = [r(t_2), \ldots, r(t_N)]^T \in \mathbb{R}^{N-1}, \end{aligned}$$

$$\begin{aligned} \mathcal{K} &= \tilde{\Omega}_2^2 - D_1\tilde{\Omega}_2^1 - D_2\tilde{\Omega}_2^0 - \tilde{\Omega}_1^2 D_1 - \tilde{\Omega}_0^2 D_2 \\ &\quad + D_1\tilde{\Omega}_1^1 D_1 + D_1\tilde{\Omega}_0^1 D_2 + D_2\tilde{\Omega}_1^0 D_1 + D_2\tilde{\Omega}_0^0 D_2, \\ h_{p_1} &= [\Omega_0^2(1, 2:N)]^T - D_1[\Omega_0^1(1, 2:N)]^T - D_2[\Omega_0^0(1, 2:N)]^T, \\ h_{p_2} &= [\Omega_1^2(1, 2:N)]^T - D_1[\Omega_1^1(1, 2:N)]^T - D_2[\Omega_1^0(1, 2:N)]^T. \end{aligned}$$

D_1 and D_2 are diagonal matrices with the elements of f_1 and f_2 on the main diagonal respectively. Note that $\mathcal{K} \in \mathbb{R}^{(N-1)\times(N-1)}$ and $h_{p_1}, h_{p_2} \in \mathbb{R}^{N-1}$. $\tilde{\Omega}_n^m = \Omega_n^m(2:N, 2:N)$ for $m, n = 0, 1, 2$.

Proof. By deriving the KKT optimality conditions and eliminating the primal variables w and e. □

The LSSVM model for the solution and its derivative in the dual form become:

$$\begin{aligned}\hat{y}(t) =& \sum_{i=2}^{N} \alpha_i \Big([\nabla_2^0 K](t_i, t) - f_1(t_i)[\nabla_1^0 K](t_i, t) - f_2(t_i)[\nabla_0^0 K](t_i, t) \Big) \\ &+ \beta_1 [\nabla_0^0 K](t_1, t) + \beta_2 [\nabla_1^0 K](t_1, t) + b, \\ \frac{d\hat{y}(t)}{dt} =& \sum_{i=2}^{N} \alpha_i \Big([\nabla_2^1 K](t_i, t) - f_1(t_i)[\nabla_1^1 K](t_i, t) - f_2(t_i)[\nabla_0^1 K](t_i, t) \Big) \\ &+ \beta_1 [\nabla_0^1 K](t_1, t) + \beta_2 [\nabla_1^1 K](t_1, t).\end{aligned}$$

BVP case:
Consider the second order boundary value problem of ODEs of the form

$$\begin{aligned} & y''(t) = f_1(t)y'(t) + f_2(t)y(t) + r(t), \quad t \in [a, c] \\ & y(a) = p_1, \quad y(c) = q_1. \end{aligned}$$

Then the parameters of the closed form approximation of the solution can be obtained by solving the following optimization problem

$$\begin{aligned} \underset{w,b,e}{\text{minimize}} \quad & \frac{1}{2} w^T w + \frac{\gamma}{2} e^T e \\ \text{subject to} \quad & w^T \varphi''(t_i) = f_1(t_i) w^T \varphi'(t_i) + \\ & f_2(t_i)[w^T \varphi(t_i) + b] + r(t_i) + e_i, \quad i = 2, ..., N-1, \\ & w^T \varphi(t_1) + b = p_1, \\ & w^T \varphi(t_N) + b = q_1. \end{aligned} \tag{3.12}$$

The same procedure can be applied to derive the Lagrangian and afterwards the KKT optimality conditions. Then one can show that the solution to the problem (3.12) is obtained by solving the following linear system

$$\left[\begin{array}{c|c|c|c} \mathcal{K}+I_{N-2}/\gamma & h_{p_1} & h_{q_1} & -f_2 \\ \hline h_{p_1}^T & 1 & \Omega_0^0(N,1) & 1 \\ \hline h_{q_1}^T & \Omega_0^0(1,N) & 1 & 1 \\ \hline -f_2^T & 1 & 1 & 0 \end{array}\right]\left[\begin{array}{c} \alpha \\ \hline \beta_1 \\ \hline \beta_2 \\ \hline b \end{array}\right]=\left[\begin{array}{c} r \\ \hline p_1 \\ \hline q_1 \\ \hline 0 \end{array}\right]$$

where

$$\alpha=[\alpha_2,\ldots,\alpha_{N-1}]^T,\ f_1=[f_1(t_2),\ldots,f_1(t_{N-1})]^T\in\mathbb{R}^{N-2},$$

$$f_2=[f_2(t_2),\ldots,f_2(t_{N-1})]^T\in\mathbb{R}^{N-2},$$

$$r=[r(t_2),\ldots,r(t_{N-1})]^T\in\mathbb{R}^{N-2},$$

$$\mathcal{K}=\tilde{\Omega}_2^2-D_1\tilde{\Omega}_2^1-D_2\tilde{\Omega}_2^0-\tilde{\Omega}_1^2D_1-\tilde{\Omega}_0^2D_2$$

$$+D_1\tilde{\Omega}_1^1D_1+D_1\tilde{\Omega}_0^1D_2+D_2\tilde{\Omega}_1^0D_1+D_2\tilde{\Omega}_0^0D_2,$$

$$h_{p_1}=[\Omega_0^2(1,2:N-1)]^T-D_1[\Omega_0^1(1,2:N-1)]^T-D_2[\Omega_0^0(1,2:N-1)]^T,$$

$$h_{q_1}=[\Omega_0^2(N,2:N-1)]^T-D_1[\Omega_0^1(N,2:N-1)]^T-D_2[\Omega_0^0(N,2:N-1)]^T.$$

D_1 and D_2 are diagonal matrices with the elements of f_1 and f_2 on the main diagonal respectively. Note that $\mathcal{K}\in\mathbb{R}^{(N-2)\times(N-2)}$ and $h_{p_1},h_{q_1}\in\mathbb{R}^{N-2}$. $\tilde{\Omega}_n^m=\Omega_n^m(2:N-1,2:N-1)$ for $m,n=0,1,2$.

The LSSVM model for the solution and its derivative are expressed in dual form as

$$\hat{y}(t)=\sum_{i=2}^{N-1}\alpha_i\Big([\nabla_2^0K](t_i,t)-f_1(t_i)[\nabla_1^0K](t_i,t)-f_2(t_i)[\nabla_0^0K](t_i,t)\Big)$$

$$+\beta_1\,[\nabla_0^0K](t_1,t)+\beta_2[\nabla_0^0K](t_N,t)+b,$$

$$\frac{d\hat{y}(t)}{dt}=\sum_{i=2}^{N-1}\alpha_i\Big([\nabla_2^1K](t_i,t)-f_1(t_i)[\nabla_1^1K](t_i,t)-f_2(t_i)[\nabla_0^1K](t_i,t)\Big)$$

$$+\beta_1[\nabla_0^1K](t_1,t)+\beta_2[\nabla_0^1K](t_N,t).$$

3.2.5 Formulation of the method for the nonlinear ODE case

In this section we formulate an optimization problem based on least squares support vector machines for solving nonlinear first order ordinary differential equations of the following form

$$y' = f(t, y), \quad y(a) = p_1, \quad a \leq t \leq c. \tag{3.13}$$

One starts with assuming the approximate solution to be of the form $\hat{y}(t) = w^T\varphi(t) + b$. Additional unknowns y_i are introduced to keep the constraints linear in w. This yields the following nonlinear optimization problem [110]:

$$\begin{aligned} \underset{w,b,e,\xi,y_i}{\text{minimize}} \quad & \frac{1}{2}w^T w + \frac{\gamma}{2}e^T e + \frac{\gamma}{2}\xi^T\xi \\ \text{subject to} \quad & w^T\varphi'(t_i) = f(t_i, y_i) + e_i, \quad i = 2, ..., N, \\ & w^T\varphi(t_1) + b = p_1, \\ & y_i = w^T\varphi(t_i) + b + \xi_i, \quad i = 2, ..., N. \end{aligned} \tag{3.14}$$

The Lagrangian of the constrained optimization problem (3.14) becomes

$$\begin{aligned} &\mathcal{L}(w, b, e_i, \xi_i, y_i, \alpha_i, \eta_i, \beta) = \\ &\frac{1}{2}w^T w + \frac{\gamma}{2}e^T e + \frac{\gamma}{2}\xi^T\xi - \sum_{i=2}^{N}\alpha_i\Big(w^T\varphi'(t_i) - \\ &f(t_i, y_i) - e_i\Big) - \beta\Big(w^T\varphi(t_1) + b - p_1\Big) - \\ &\sum_{i=2}^{N}\eta_i\Big(y_i - w^T\varphi(t_i) - b - \xi_i\Big). \end{aligned}$$

After obtaining KKT optimality conditions, and elimination of the primal variables w, $\{e_i\}_{i=2}^N$ and $\{\xi_i\}_{i=2}^N$ and making use of Mercer's Theorem, the solution is obtained in the dual by solving the following nonlinear system of equations

$$\left[\begin{array}{c|c|c|c|c} \widehat{\Omega_1^1} & \tilde{\Omega}_0^1 & h_1^T & 0_{N-1} & 0_{(N-1)\times(N-1)} \\ \hline (\tilde{\Omega}_0^1)^T & \widehat{\Omega}_0^0 & h_0^T & 1_{N-1} & -I_{N-1} \\ \hline h_1 & h_0 & \Omega_0^0(1,1) & 1 & 0_{N-1}^T \\ \hline 0_{N-1}^T & 1_{N-1}^T & 1 & 0 & 0_{N-1}^T \\ \hline D(y) & I_{N-1} & 0_{N-1} & 0_{N-1} & 0_{(N-1)\times(N-1)} \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \eta \\ \hline \beta \\ \hline b \\ \hline y \end{array}\right] = \left[\begin{array}{c} f(y) \\ \hline 0_{N-1} \\ \hline p_1 \\ \hline 0 \\ \hline 0_{N-1} \end{array}\right] \tag{3.15}$$

where

$$\widehat{\Omega}_1^1 = \tilde{\Omega}_1^1 + I_{N-1}/\gamma, \quad \widehat{\Omega}_0^0 = \tilde{\Omega}_0^0 + I_{N-1}/\gamma,$$

$$D(y) = \mathrm{diag}(f'(y))$$

$$f(y) = [f(t_2, y_2), \ldots, f(t_N, y_N)]^T,$$

$$f'(y) = [\frac{\partial f(t,y)}{\partial y}\Big|_{t=t_2, y=y_2}, \ldots, \frac{\partial f(t,y)}{\partial y}\Big|_{t=t_N, y=y_N}],$$

$$\alpha = [\alpha_2, \ldots, \alpha_N]^T, \ \eta = [\eta_2, \ldots, \eta_N]^T,$$

$$y = [y_2, \ldots, y_N]^T, \ \tilde{\Omega}_0^0 = \Omega_0^0(2:N, 2:N),$$

$$\tilde{\Omega}_1^1 = \Omega_1^1(2:N, 2:N), \ \tilde{\Omega}_0^1 = \Omega_0^1(2:N, 2:N),$$

$$h_0 = \left[\Omega_0^0(1,2), \ldots, \Omega_0^0(1,N)\right],$$

$$h_1 = \left[\Omega_0^1(1,2), \ldots, \Omega_0^1(1,N)\right], 0_{N-1} = [0, \ldots, 0]^T \in \mathbb{R}^{N-1}.$$

The nonlinear system (3.15), which consists of $3N-1$ equations with $3N-1$ unknowns $(\alpha, \eta, \beta, b, y)$, is solved by Newton's method. The model in the dual form becomes

$$\hat{y}(t) = \sum_{i=2}^{N} \alpha_i [\nabla_1^0 K](t_i, t) + \sum_{i=2}^{N} \eta_i [\nabla_0^0 K](t_i, t) + \beta\, [\nabla_0^0 K](t_1, t) + b,$$

where $\nabla_0^0 K(t,s) = K(t,s)$ is the kernel function.

3.2.6 Solution on a long time interval

Consider now the situation where a given differential equation has to be solved for a large time interval $[a, c]$. It should be noted that in order to improve the

accuracy (or maintain the same order of accuracy on the whole domain) we then need to increase the number of collocation points. This approach however leads to a larger system of equations.

In order to implement the proposed method for solving problems involving large time intervals efficiently, the time windowing technique is applied [162]. At first the interval $\Xi = [a, c]$ is decomposed into S sub-intervals as $\Xi = \bigcup_{k=1}^{S} \Xi_k$. We assume that the approximate solution on the k-th sub-interval has the form $\hat{y}_k(t) = w_k^T \varphi(t) + b_k$. Then the problem is solved in each sub-interval Ξ_k using the described method in previous sections. The computed approximate solution at the final point in the sub-interval Ξ_k is used as starting point (initial value) for the consecutive sub-interval Ξ_{k+1}.

Utilizing this approach will result in solving S smaller systems of equations, which is computationally more efficient than solving a very large system of equations obtained by considering the whole domain Ξ (with the same total number of collocation points). The procedure is outlined in Algorithm 2.

Algorithm 2: Approximating the solution on a large interval

1: Decompose the domain $\Xi = [a, c]$ into S sub-intervals.
2: set $\Gamma = (c - a)/S$, $t_{in} := a$, $y_{in} := p_1$, $t_f := t_{in} + \Gamma$.
3: **for** $k = 1$ **to** S **do**
4: Obtain a LSSVM model for the k-th sub-interval $[t_{in}, t_f]$ i.e. $\hat{y}_k(t) = w_k^T \varphi_k(t) + b_k$.
5: set $t_{in} := t_f$, $y_{in} := \hat{y}(t_f)$, $t_f := t_{in} + \Gamma$
6: **end for**
7: For a given test point t:

- Check to which sub-interval it belongs,
- Use the corresponding model to compute the approximate solution at the given point.

3.3 Learning the Solution of DAEs

This section presents the formulation of the method for initial and boundary value problems in differential algebraic equations.

3.3.1 Formulation of the method for IVPs in DAEs

Consider linear time varying initial value problem in DAEs of the following from

$$\begin{aligned} &Z(t)\dot{X}(t) = A(t)X(t) + B(t)u(t),\ t \in [t_{in}, t_f], \\ &X(t_{in}) = X_0, \end{aligned} \tag{3.16}$$

where $Z(t) = [z_{ij}(t)]$, $A(t) = [a_{ij}(t)] \in \mathbb{R}^{m\times m}$ and $B(t) \in \mathbb{R}^{m\times r}$. The state vector $X = [x_1, ..., x_m]^T \in \mathbb{R}^m$, the input vector $u \in \mathbb{R}^r$ and $\dot{X}(t) = \frac{dX}{dt}$. $Z(t)$ may be singular on $[t_{in}, t_f]$ with variable rank and the DAE may have an index that is larger than one. When Z is nonsingular, equation (3.16) can be converted to an equivalent explicit ODE system. In addition we assume that $Z(t)$, $A(t)$ and $B(t)u(t)$ are sufficiently smooth and the DAE (3.16) is solvable. (see Definition 2.1 in [49] and references therein for a more detailed discussion about solvability).

Let us assume that a general approximate solution to i-th equation of (3.16) is of the form of $\hat{x_i}(t) = w_i^T\varphi(t)+b_i$, where w_i and b_i are parameters of the model that have to be determined. To obtain the optimal value of these parameters, collocation methods can be used which assume a discretization of the interval $[t_{in}, t_f]$ into a set of collocation points $\Upsilon = \{t_{in} = t_1 < t_2 < ... < t_N = t_f\}$. Therefore the adjustable parameters w_i and b_i, for $i = 1, ..., m$, are to be found by solving the following optimization problem:

$$\begin{aligned} \underset{\hat{X}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{N}\left[(Z\hat{X}' - A\hat{X} - Bu)(t_i)\right]^2 \\ \text{subject to} \quad & \hat{X}(t_{in}) = X_0, \end{aligned} \tag{3.17}$$

where N is the number of collocation points (which is equal to the number of training points) used to undertake the learning process. For simplicity let us assume that $X_0 = [p_1; ...; p_m]$ and $g(t) = B(t)u(t)$. In what follows we formulate the optimization problem in the LSSVM framework for solving linear time varying differential algebraic equations.

Remark 3.3.1. *In order to keep the notations as simple as possible, we utilized the same feature map $\varphi(t)$ for all the states, i.e. $\hat{x_i}(t) = w_i^T\varphi(t) + b_i$. Nevertheless it is possible to use different mapping functions, as long as the corresponding kernel functions satisfy the Mercer's Theorem [167], and then one has more hyper-parameters to tune.*

3.3.1.1 Singular ODE System

Let us consider DAEs (3.16). The approximate solution, $\hat{x}_i(t) = w_i^T \varphi(t) + b_i$, for $i = 1, ..., m$ is then obtained by solving the following optimization problem

$$
\begin{aligned}
\underset{w_i, b_i, e_\ell^i}{\text{minimize}} \quad & \frac{1}{2} \sum_{\ell=1}^{m} w_\ell^T w_\ell + \frac{\gamma}{2} \sum_{\ell=1}^{m} e_\ell^T e_\ell \\
\text{subject to} \quad & \begin{bmatrix} z_{11}(t_i) & \cdots & z_{1m}(t_i) \\ \vdots & \ddots & \vdots \\ z_{m1}(t_i) & \cdots & z_{mm}(t_i) \end{bmatrix} \begin{bmatrix} w_1^T \varphi'(t_i) \\ \vdots \\ w_m^T \varphi'(t_i) \end{bmatrix} = \\
& \begin{bmatrix} a_{11}(t_i) & \cdots & a_{1m}(t_i) \\ \vdots & \ddots & \vdots \\ a_{m1}(t_i) & \cdots & a_{mm}(t_i) \end{bmatrix} \begin{bmatrix} w_1^T \varphi(t_i) + b_1 \\ \vdots \\ w_m^T \varphi(t_i) + b_m \end{bmatrix} + \\
& \begin{bmatrix} g_1(t_i) \\ \vdots \\ g_m(t_i) \end{bmatrix} + \begin{bmatrix} e_1(t_i) \\ \vdots \\ e_m(t_i) \end{bmatrix}, \text{for } i = 2, ..., N, \\
& \begin{bmatrix} w_1^T \varphi(t_1) + b_1 \\ \vdots \\ w_m^T \varphi(t_1) + b_m \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_m \end{bmatrix}.
\end{aligned} \tag{3.18}
$$

Lemma 3.3.1. *Given a positive definite kernel function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with $K(t, s) = \varphi(t)^T \varphi(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (3.18) is given by the following dual problem [116]:*

$$
\left[\begin{array}{c|c|c} \mathcal{K} & \mathcal{S} & -F_A \\ \hline \mathcal{S}^T & \Omega_0^0(1,1) \times I_{m \times m} & I_{m \times m} \\ \hline -F_A^T & I_{m \times m} & 0_{m \times m} \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \\ \hline b \end{array}\right] = \left[\begin{array}{c} G \\ \hline P \\ \hline 0 \end{array}\right] \tag{3.19}
$$

with

$$\alpha = \left[\begin{array}{c} \alpha^1 \\ \hline \vdots \\ \hline \alpha^m \end{array}\right] = [\alpha_2^1; ...; \alpha_N^1; \cdots; \alpha_2^m; ...; \alpha_N^m] \in \mathbb{R}^{m(N-1)},$$

$$\beta = [\beta_1; \cdots; \beta_m],\ b = [b_1; ...; b_m],\ P = [p_1; ...; p_m],$$

$$G = [g_1(t_2); \cdots; g_1(t_N); \cdots; g_m(t_2); \cdots; g_m(t_N)] \in \mathbb{R}^{m(N-1)},$$

$$F_A = \begin{bmatrix} F_{A_{11}} & \dots & F_{A_{1m}} \\ \vdots & & \vdots \\ F_{A_{m1}} & \dots & F_{A_{mm}} \end{bmatrix} \in \mathbb{R}^{m(N-1)\times m},$$

$$F_{A_{kl}} = [a_{kl}(t_2); \cdots; a_{kl}(t_N)] \in \mathbb{R}^{N-1},\ \textit{for } k, l = 1, ..., m$$

$$F_{Z_{kl}} = [z_{kl}(t_2); \cdots; z_{kl}(t_N)] \in \mathbb{R}^{N-1},\ \textit{for } k, l = 1, ..., m$$

$$\mathcal{K} = \begin{bmatrix} \mathcal{K}_{11} & \dots & \mathcal{K}_{1m} \\ \vdots & & \vdots \\ \mathcal{K}_{m1} & \dots & \mathcal{K}_{mm} \end{bmatrix} \in \mathbb{R}^{m(N-1)\times m(N-1)},$$

$$\mathcal{K}_{ii} = \overline{D_{Z_i}}\bar{\Omega}_1^1\overline{D_{Z_i}}^T - \overline{D_{A_i}}\bar{\Omega}_1^0\overline{D_{Z_i}}^T - \overline{D_{Z_i}}\bar{\Omega}_0^1\overline{D_{A_i}}^T +$$

$$\overline{D_{A_i}}\bar{\Omega}_0^0\overline{D_{A_i}}^T + I_{N-1}/\gamma,\ i = 1, ..., m,$$

$$\mathcal{K}_{ij} = \overline{D_{Z_i}}\bar{\Omega}_1^1\overline{D_{Z_j}}^T - \overline{D_{A_i}}\bar{\Omega}_1^0\overline{D_{Z_j}}^T - \overline{D_{Z_i}}\bar{\Omega}_0^1\overline{D_{A_j}}^T +$$

$$\overline{D_{A_i}}\bar{\Omega}_0^0\overline{D_{A_j}}^T,\ i, j = 1, ..., m \textit{ and } i \neq j,$$

$$\mathcal{S} = \begin{bmatrix} \mathcal{S}_{11} & \dots & \mathcal{S}_{1m} \\ \vdots & & \vdots \\ \mathcal{S}_{m1} & \dots & \mathcal{S}_{mm} \end{bmatrix} \in \mathbb{R}^{m(N-1)\times m},$$

$$\mathcal{S}_{ij} = D_{Z_{ij}}\Omega_0^1(1,:)^T - D_{A_{ij}}\Omega_0^0(1,:)^T,\ i, j = 1, ..., m,$$

$$\overline{D_{Z_i}} = [D_{Z_{i1}}, ..., D_{Z_{im}}], \overline{D_{A_i}} = [D_{A_{i1}}, ..., D_{A_{im}}],$$

$$\bar{\Omega}_l^k = \begin{bmatrix} \tilde{\Omega}_k^l & & \\ & \ddots & \\ & & \tilde{\Omega}_k^l \end{bmatrix} \in \mathbb{R}^{m(N-1)\times m(N-1)}, k, l = 0, 1.$$

$D_{A_{ij}}$ and $D_{Z_{ij}}$ are diagonal matrices with the elements of $F_{A_{ij}}$ and $F_{Z_{ij}}$ on the main diagonal respectively. $\Omega_k^l(1,:) = [\Omega_k^l(1,2);...;\Omega_k^l(1,N)]^T$ and $\tilde{\Omega}_k^l = \Omega_k^l(2:N, 2:N)$ for $k,l = 0,1$. Also note that $\mathcal{K} = \mathcal{K}^T$.

Proof. Consider the Lagrangian of problem (3.18):

$$\mathcal{L}(w_\ell, b_\ell, e_\ell^i, \alpha_i^\ell, \beta_\ell) =$$

$$\frac{1}{2}\sum_{\ell=1}^{m} w_\ell^T w_\ell + \frac{\gamma}{2}\sum_{\ell=1}^{m} e_\ell^T e_\ell - \sum_{\ell=1}^{m}\Bigg[\sum_{i=2}^{N}\alpha_i^\ell\Bigg[w_\ell^T\Big(z_{\ell\ell}(t_i)\varphi'(t_i) -$$

$$a_{\ell\ell}(t_i)\varphi(t_i)\Big) - \sum_{\substack{k=1\\k\neq\ell}}^{m} w_k^T\Big(z_{\ell k}(t_i)\varphi'(t_i) - a_{\ell k}(t_i)\varphi(t_i)\Big) -$$

$$\sum_{k=1}^{m} a_{\ell k}(t_i)b_k - e_\ell^i - g_\ell^i\Bigg]\Bigg] - \sum_{\ell=1}^{m}\beta_\ell\Big(w_\ell^T\varphi(t_1) + b_\ell - p_\ell\Big)$$

where $\{\alpha_i^\ell\}_{i=2}^N$, for $\ell = 1,...,m$ and $\{\beta_\ell\}_{\ell=1}^m$ are Lagrange multipliers. $g_\ell^i = g_\ell(t_i)$, $e_\ell^i = e_\ell(t_i)$ for $i = 2,...,N$ and $\ell = 1,...,m$. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\frac{\partial\mathcal{L}}{\partial w_\ell} = 0 \rightarrow\ w_\ell = \sum_{v=1}^{m}\sum_{i=2}^{N}\alpha_i^v\Big(z_{v\ell}(t_i)\varphi'(t_i) - a_{v\ell}(t_i)\varphi(t_i)\Big) +$$

$$\beta_\ell\varphi(t_1),\ \ell = 1,...,m,$$

$$\frac{\partial\mathcal{L}}{\partial b_\ell} = 0 \rightarrow \sum_{k=1}^{m}\sum_{i=2}^{N}\alpha_i^k a_{k\ell}(t_i) - \beta_\ell = 0,\ \ell = 1,...,m,$$

$$\frac{\partial\mathcal{L}}{\partial e_\ell^i} = 0 \rightarrow\ e_\ell^i = -\frac{\alpha_i^\ell}{\gamma},\quad i = 2,...,N,\ \ell = 1,...,m,$$

$$\frac{\partial\mathcal{L}}{\partial \alpha_i^\ell} = 0 \rightarrow \sum_{k=1}^{m} w_k^T\Big(z_{\ell k}(t_i)\varphi'(t_i) - a_{\ell k}(t_i)\varphi(t_i)\Big)$$

$$-\sum_{k=1}^{m} a_{\ell k}(t_i)b_k - e_\ell^i = g_\ell^i,\quad i = 2,...,N,\ \ell = 1,...,m$$

$$\frac{\partial\mathcal{L}}{\partial \beta_\ell} = 0 \rightarrow\ w_\ell^T\varphi(t_1) + b_\ell = p_\ell,\ \ell = 1,...,m.$$

Eliminating $\{w_\ell\}_{\ell=1}^m$ and $\left\{e_\ell^i\right\}_{i=2}^N$ for $\ell = 1, ..., m$ from the corresponding KKT optimality conditions yields the following set of equations

$$
\begin{cases}
g_\ell^i = \sum_{v=1}^{m} \Bigg[\sum_{j=2}^{N} \alpha_j^v \Bigg[\sum_{k=1}^{m} z_{\ell k}(t_i)\Omega_1^1(j,i) z_{vk}(t_j) - \\
\sum_{k=1}^{m} a_{\ell k}(t_i)\Omega_1^0(j,i) z_{vk}(t_j) - \sum_{k=1}^{m} z_{\ell k}(t_i)\Omega_0^1(j,i) a_{vk}(t_j) + \\
\sum_{k=1}^{m} a_{\ell k}(t_i)\Omega_0^0(j,i) a_{vk}(t_j) \Bigg] \Bigg] + \\
\sum_{v=1}^{m} \beta_v \Big(z_{\ell v}(t_i)\Omega_0^1(1,i) - a_{\ell v}(t_i)\Omega_0^0(1,i) \Big) \\
+ \frac{\alpha_i^\ell}{\gamma} - \sum_{k=1}^{m} a_{\ell k}(t_i) b_k, \quad i = 2, ..., N, \text{ and } \ell = 1, ..., m, \\
p_\ell = \sum_{v=1}^{m} \Bigg[\sum_{j=2}^{N} \alpha_j^v \Big(z_{v\ell}(t_j)\Omega_1^0(j,1) - a_{v\ell}(t_j)\Omega_0^0(j,1) \Big) \Bigg] \\
+ \beta_\ell\, \Omega_0^0(1,1) + b_\ell, \ \ell = 1, ..., m, \\
0 = \sum_{v=1}^{m} \sum_{j=2}^{N} \alpha_j^v \ a_{v\ell}(t_j) - \beta_\ell, \ \ell = 1, ..., m.
\end{cases}
$$

Finally writing these equations in matrix form will result in the linear system (3.19). □

The model in the dual form becomes

$$
\begin{aligned}
\hat{x}_\ell(t) = & \sum_{v=1}^{m} \sum_{i=2}^{N} \alpha_i^v \Big(z_{v\ell}(t_i)[\nabla_1^0 K](t_i, t) - a_{v\ell}(t_i)[\nabla_0^0 K](t_i, t) \Big) + \\
& \beta_\ell\, [\nabla_0^0 K](t_1, t) + b_\ell, \ \ell = 1, ..., m,
\end{aligned}
$$

where K is the kernel function.

3.3.1.2 Explicit ODE System

Let us consider the system of IVPs (3.16) with Z as identity matrix as a special case. This results in an ODE system.

Corollary 3.3.1. *Given a positive definite kernel function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with $K(t,s) = \varphi(t)^T\varphi(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (3.18) with the matrix $Z = I$, is then given by the following dual problem [116]:*

$$\left[\begin{array}{c|c|c} \mathcal{K} & \mathcal{S} & -F_A \\ \hline \mathcal{S}^T & \Omega_0^0(1,1) \times I_{m\times m} & I_{m\times m} \\ \hline -F_A^T & I_{m\times m} & 0_{m\times m} \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \\ \hline b \end{array}\right] = \left[\begin{array}{c} G \\ \hline P \\ \hline 0 \end{array}\right] \tag{3.20}$$

where α, β, b, R, P and F_A are defined as in Eq. (3.19). Also with

$$F_{A_{kl}} = [a_{kl}(t_2); \cdots ; a_{kl}(t_N)] \in \mathbb{R}^{N-1}, \text{ for } k, l = 1, ..., m$$

$$\mathcal{K} = \begin{bmatrix} \mathcal{K}_{11} & \dots & \mathcal{K}_{1m} \\ \vdots & & \vdots \\ \mathcal{K}_{m1} & \dots & \mathcal{K}_{mm} \end{bmatrix} \in \mathbb{R}^{m(N-1)\times m(N-1)},$$

$$\mathcal{K}_{ii} = \tilde{\Omega}_1^1 - D_{ii}\tilde{\Omega}_1^0 - \tilde{\Omega}_0^1 D_{ii} + T_{ii} + I_{N-1}/\gamma, \ i = 1, ..., m,$$

$$\mathcal{K}_{ij} = -\tilde{\Omega}_0^1 D_{ji} - D_{ij}\tilde{\Omega}_1^0 + T_{ij}, \ i, j = 1, ..., m \text{ and } i \neq j,$$

$$\mathcal{S} = \begin{bmatrix} \mathcal{S}_{11} & \dots & \mathcal{S}_{1m} \\ \vdots & & \vdots \\ \mathcal{S}_{m1} & \dots & \mathcal{S}_{mm} \end{bmatrix} \in \mathbb{R}^{m(N-1)\times m},$$

$$\mathcal{S}_{ii} = \Omega_0^1(1,:)^T - D_{ii}\Omega_0^0(1,:)^T, \ i = 1, ..., m,$$

$$\mathcal{S}_{ij} = -D_{ij}\Omega_0^0(1,:)^T, i, j = 1, ..., m \text{ and } i \neq j,$$

$$T_{ij} = \bar{D}_i \bar{\Omega} \bar{D}_j{}^T \in \mathbb{R}^{(N-1)\times(N-1)} \ i, j = 1, ..., m,$$

$$\bar{D}_i = [D_{i1}, ..., D_{im}], \bar{D}_j = [D_{j1}, ..., D_{jm}],$$

$$\bar{\Omega} = \begin{bmatrix} \tilde{\Omega}_0^0 & & \\ & \ddots & \\ & & \tilde{\Omega}_0^0 \end{bmatrix} \in \mathbb{R}^{m(N-1)\times m(N-1)}$$

D_{ij} is a diagonal matrix with the elements of F_{ij} on the main diagonal. $\Omega_n^m(1,:) = [\Omega_n^m(1,2); ...; \Omega_n^m(1,N)]^T$ and $\tilde{\Omega}_n^m = \Omega_n^m(2:N, 2:N)$ for $n, m = 0, 1$. Also note that $\mathcal{K} = \mathcal{K}^T$.

The model in the dual form becomes

$$\hat{x}_\ell(t) = \sum_{i=2}^{N} \alpha_i^\ell \Big([\nabla_1^0 K](t_i, t) - a_{\ell\ell}(t_i)[\nabla_0^0 K](t_i, t) \Big) - \sum_{\substack{v=1 \\ v\neq\ell}}^{m} \sum_{i=2}^{N} \alpha_i^v \Big(a_{v\ell}(t_i)[\nabla_0^0 K](t_i, t) \Big)$$

$$+ \beta_\ell \, [\nabla_0^0 K](t_1, t) + b_\ell, \ \ell = 1, ..., m,$$

where K is the kernel function.

3.3.2 Formulation of the method for BVPs in DAEs

Consider linear time varying boundary value problem in DAEs of the following from

$$\begin{aligned} &Z(t)\dot{X}(t) = A(t)X(t) + g(t), \ t \in [t_{in}, t_f], \\ &FX(t_{in}) + HX(t_f) = X_0, \end{aligned} \tag{3.21}$$

where matrices $Z(t)$, $A(t)$ and the state vector $X(t)$ are defined as in Eq. (3.16). $F = [f_{ij}]$ and $H = [h_{ij}] \in \mathbb{R}^{m\times m}$. The input is $g(t)$ and $\dot{X}(t) = \frac{dX}{dt}$. $Z(t)$ may be singular on $[t_{in}, t_f]$ with variable rank and the DAE may have an index that is larger than one. As before we assume that $Z(t)$, $A(t)$ and $g(t)$ are sufficiently smooth and the DAE (3.21) is solvable. When Z is nonsingular, equation (3.21) can be converted to equivalent explicit ODE system.

The approximate solution, $\hat{x}_i(t) = w_i^T \varphi(t) + b_i$, for $i = 1, ..., m$ is then obtained by solving the following optimization problem,

$$
\begin{aligned}
&\underset{w_i,b_i,e_\ell^i}{\text{minimize}} \quad \frac{1}{2}\sum_{\ell=1}^{m} w_\ell^T w_\ell + \frac{\gamma}{2}\sum_{\ell=1}^{m} e_\ell^T e_\ell + \frac{\zeta}{2}\sum_{\ell=1}^{m} \xi_\ell^2 \\
&\text{subject to} \quad \begin{bmatrix} z_{11}(t_i) & \cdots & z_{1m}(t_i) \\ \vdots & \ddots & \vdots \\ z_{m1}(t_i) & \cdots & z_{mm}(t_i) \end{bmatrix} \begin{bmatrix} w_1^T\varphi'(t_i) \\ \vdots \\ w_m^T\varphi'(t_i) \end{bmatrix} = \\
&\qquad \begin{bmatrix} a_{11}(t_i) & \cdots & a_{1m}(t_i) \\ \vdots & \ddots & \vdots \\ a_{m1}(t_i) & \cdots & a_{mm}(t_i) \end{bmatrix} \begin{bmatrix} w_1^T\varphi(t_i) + b_1 \\ \vdots \\ w_m^T\varphi(t_i) + b_m \end{bmatrix} + \\
&\qquad \begin{bmatrix} g_1(t_i) \\ \vdots \\ g_m(t_i) \end{bmatrix} + \begin{bmatrix} e_1(t_i) \\ \vdots \\ e_m(t_i) \end{bmatrix}, \text{for } i = 2, ..., N-1, \\
&\qquad \begin{bmatrix} f_{11} & \cdots & f_{1m} \\ \vdots & \ddots & \vdots \\ f_{m1} & \cdots & f_{mm} \end{bmatrix} \begin{bmatrix} w_1^T\varphi(t_1) + b_1 \\ \vdots \\ w_m^T\varphi(t_1) + b_m \end{bmatrix} + \\
&\qquad \begin{bmatrix} h_{11} & \cdots & h_{1m} \\ \vdots & \ddots & \vdots \\ h_{m1} & \cdots & h_{mm} \end{bmatrix} \begin{bmatrix} w_1^T\varphi(t_N) + b_1 \\ \vdots \\ w_m^T\varphi(t_N) + b_m \end{bmatrix} = \\
&\qquad \begin{bmatrix} p_1 \\ \vdots \\ p_m \end{bmatrix} + \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_m \end{bmatrix}.
\end{aligned} \tag{3.22}
$$

Lemma 3.3.2. *Given a positive definite kernel function $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with $K(t,s) = \varphi(t)^T\varphi(s)$ and a regularization constant $\gamma, \xi \in \mathbb{R}^+$, the solution to (3.22) is given by the following dual problem [116]:*

$$
\left[\begin{array}{c|c|c} \mathcal{K} & \mathcal{U} & -F_A \\ \hline \mathcal{U}^T & \Delta & \Pi \\ \hline -F_A^T & \Pi^T & 0_{m\times m} \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \\ \hline b \end{array}\right] = \left[\begin{array}{c} G \\ \hline P \\ \hline 0 \end{array}\right] \tag{3.23}
$$

with

$$\alpha = \left[\begin{array}{c} \alpha^1 \\ \hline \vdots \\ \hline \alpha^m \end{array}\right] = [\alpha_2^1; ...; \alpha_{N-1}^1; \cdots; \alpha_2^m; ...; \alpha_{N-1}^m] \in \mathbb{R}^{m(N-2)},$$

$$\beta = [\beta_1; \cdots; \beta_m],\ b = [b_1; ...; b_m],\ P = [p_1; ...; p_m],$$

$$G = [g_1(t_2); \cdots; g_1(t_{N-1}); \cdots; g_m(t_2); \cdots; g_m(t_{N-1})] \in \mathbb{R}^{m(N-2)},$$

$$F_A = \left[\begin{array}{ccc} F_{A_{11}} & \dots & F_{A_{1m}} \\ \vdots & & \vdots \\ F_{A_{m1}} & \dots & F_{A_{mm}} \end{array}\right] \in \mathbb{R}^{m(N-2)\times m},$$

$$F_{A_{kl}} = [a_{kl}(t_2); \cdots; a_{kl}(t_{N-1})] \in \mathbb{R}^{N-2},\ \textit{for } k, l = 1, ..., m,$$

$$F_{Z_{kl}} = [z_{kl}(t_2); \cdots; z_{kl}(t_{N-1})] \in \mathbb{R}^{N-2},\ \textit{for } k, l = 1, ..., m,$$

$$\Pi = F + H,$$

$$\Delta = \left(FF^T\right)\Omega_0^0(1,1) + \left(FH^T + HF^T\right)\Omega_0^0(1,N) + \left(HH^T\right)\Omega_0^0(N,N)$$

$$+ I_m/\zeta,$$

$$\mathcal{K} = \left[\begin{array}{ccc} \mathcal{K}_{11} & \dots & \mathcal{K}_{1m} \\ \vdots & & \vdots \\ \mathcal{K}_{m1} & \dots & \mathcal{K}_{mm} \end{array}\right] \in \mathbb{R}^{m(N-2)\times m(N-2)},$$

$$\mathcal{K}_{ii} = \overline{D_{Z_i}}\bar{\Omega}_1^1\overline{D_{Z_i}}^T - \overline{D_{A_i}}\bar{\Omega}_1^0\overline{D_{Z_i}}^T - \overline{D_{Z_i}}\bar{\Omega}_0^1\overline{D_{A_i}}^T +$$

$$\overline{D_{A_i}}\bar{\Omega}_0^0\overline{D_{A_i}}^T + I_{N-1}/\gamma,\ i = 1, ..., m,$$

$$\mathcal{K}_{ij} = \overline{D_{Z_i}}\bar{\Omega}_1^1\overline{D_{Z_j}}^T - \overline{D_{A_i}}\bar{\Omega}_1^0\overline{D_{Z_j}}^T - \overline{D_{Z_i}}\bar{\Omega}_0^1\overline{D_{A_j}}^T +$$

$$\overline{D_{A_i}}\bar{\Omega}_0^0\overline{D_{A_j}}^T,\ i, j = 1, ..., m\ \textit{and}\ i \neq j,$$

$$\mathcal{U} = \left[\begin{array}{ccc} \mathcal{U}_{11} & \dots & \mathcal{U}_{1m} \\ \vdots & & \vdots \\ \mathcal{U}_{m1} & \dots & \mathcal{U}_{mm} \end{array}\right] \in \mathbb{R}^{m(N-2)\times m},$$

$$\mathcal{U}_{ij} = \Big(\sum_{k=1}^{m} D_{Z_{ik}} f_{jk}\Big)\Omega_0^1(1,:)^T + \Big(\sum_{k=1}^{m} D_{Z_{ik}} h_{jk}\Big)\Omega_0^1(N,:)^T$$

$$- \Big(\sum_{k=1}^{m} D_{A_{ik}} f_{jk}\Big)\Omega_0^0(1,:)^T - \Big(\sum_{k=1}^{m} D_{A_{ik}} h_{jk}\Big)\Omega_0^0(N,:)^T,\ i,j = 1,...,m,$$

$$\overline{D_{Z_i}} = [D_{Z_{i1}},...,D_{Z_{im}}], \overline{D_{A_i}} = [D_{A_{i1}},...,D_{A_{im}}],$$

$$\bar{\Omega}_l^k = \begin{bmatrix} \tilde{\Omega}_k^l & & \\ & \ddots & \\ & & \tilde{\Omega}_k^l \end{bmatrix} \in \mathbb{R}^{m(N-2)\times m(N-2)}, k,l = 0,1.$$

$D_{A_{ij}}$ and $D_{Z_{ij}}$ are diagonal matrices with the elements of $F_{A_{ij}}$ and $F_{Z_{ij}}$ on the main diagonal respectively. $\Omega_k^l(1,:) = [\Omega_k^l(1,2);...;\Omega_k^l(1,N-1)]^T$, $\Omega_k^l(N,:) = [\Omega_k^l(N,2);...;\Omega_k^l(N,N-1)]^T$ and $\tilde{\Omega}_k^l = \Omega_k^l(2:N-1,2:N-1)$ for $k,l = 0,1$. Also note that $\mathcal{K} = \mathcal{K}^T$.

Proof. By deriving the KKT optimality conditions and eliminating the primal variables w and e. □

The model in the dual form becomes

$$\hat{x}_\ell(t) = \sum_{v=1}^{m}\sum_{i=2}^{N-1} \alpha_i^v \Big(z_{v\ell}(t_i)[\nabla_1^0 K](t_i,t) - a_{v\ell}(t_i)[\nabla_0^0 K](t_i,t)\Big) +$$

$$\sum_{v=1}^{m} \beta_v \Big([\nabla_0^0 K](t_1,t) f_{v\ell} + [\nabla_0^0 K](t_N,t) h_{v\ell}\Big) + b_\ell,\ \ell = 1,...,m,$$

where K is the kernel function.

Remark 3.3.2. *A singular system with a discontinuous input will exhibit a jump. The LSSVM approximation with Gaussian kernel (which provides a smooth approximation) shows a spurious oscillation near the discontinuity. This oscillation behavior is a common phenomenon known as the Gibbs phenomenon that appears when the underlying function being approximated has jump discontinuities. Some methods have been suggested in the literature to reduce the effect of Gibbs phenomenon (see [78]). Another approach is to use a continuous approximation of the non smooth input signal as the new input for the system [136].*

Remark 3.3.3. *Concerning practical application of the proposed method for finding the approximate solution to the given DAEs on a very long time interval, the approach described in Section 3.2.6 can be utilized here as well.*

Remark 3.3.4. *In general, but not necessarily, one starts with equidistant mesh points. The residual on the validation set is then monitored and for the subintervals which the error is not sufficiently small one may refine the mesh in order to achieve adequate accuracy. The residual at a given point t_i can be computed as*

$$e(t_i) = Z(t_i)\dot{X}(t_i) - A(t_i)X(t_i) - g(t_i). \tag{3.24}$$

3.4 Learning the Solution of PDEs

First some of the operators that are going to be used in the subsequent subsections are defined. Basically what we need is the generalization of the operators defined in subsection 3.2.2 for d-dimensional ($d > 1$) input space. Without loss of generality let us assume that $d = 2$ i.e. the training points $z_i \in \mathbb{R}^2$. Suppose that $z_1 = (x_1, t_1)^T$ and $z_2 = (x_2, t_2)^T$ are two arbitrary points in $\mathbb{R}^2$ (x, t-coordinates). Then let us define the following differential operator which will be used in subsequent sections

$$\nabla_{s^{(n)},\, p^{(m)}} \equiv \frac{\partial^{n+m}}{\partial s^n \partial p^m}. \tag{3.25}$$

If $\varphi(z_1)^T\varphi(z_2) = K(z_1, z_2)$, then one can show that

$$\begin{aligned}
\Big[\varphi_{x^{(n)}}(z_1)\Big]^T\varphi_{x^{(m)}}(z_2) &= \nabla_{x_1^{(n)},\, x_2^{(m)}}\Big[K(z_1, z_2)\Big] = \frac{\partial^{n+m}K(z_1, z_2)}{\partial {x_1}^n \partial {x_2}^m},\\
\Big[\varphi_{x^{(n)}}(z_1)\Big]^T\varphi_{t^{(m)}}(z_2) &= \nabla_{x_1^{(n)},\, t_2^{(m)}}\Big[K(z_1, z_2)\Big] = \frac{\partial^{n+m}K(z_1, z_2)}{\partial {x_1}^n \partial {t_2}^m},\\
\Big[\varphi_{t^{(n)}}(z_1)\Big]^T\varphi_{t^{(m)}}(z_2) &= \nabla_{t_1^{(n)},\, t_2^{(m)}}\Big[K(z_1, z_2)\Big] = \frac{\partial^{n+m}K(z_1, z_2)}{\partial {t_1}^n \partial {t_2}^m},\\
\Big[\varphi_{t^{(n)}}(z_1)\Big]^T\varphi_{x^{(m)}}(z_2) &= \nabla_{t_1^{(n)},\, x_2^{(m)}}\Big[K(z_1, z_2)\Big] = \frac{\partial^{n+m}K(z_1, z_2)}{\partial {t_1}^n \partial {x_2}^m}.
\end{aligned} \tag{3.26}$$

Here $\varphi_{x^{(n)}}$ and $\varphi_{t^{(n)}}$ are the n-th derivative of the feature map φ with respect to variable x and t respectively. Note that if either m or n is zero, we do not take the derivative of the term w.r.t to the corresponding variable. More precisely suppose $m = 0$ then we use the following notations:

$$\Big[\varphi_{x^{(n)}}(z_1)\Big]^T\varphi_{x^{(0)}}(z_2) = \nabla_{x_1^{(n)},\, x_2^{(0)}}\Big[K(z_1,z_2)\Big] = \nabla_{x_1^{(n)},0}\Big[K(z_1,z_2)\Big] = \frac{\partial^n K(z_1,z_2)}{\partial x_1^n}.$$

For instance if K is chosen to be the RBF kernel

$$K(z_1,z_2) = \exp(-\frac{\|z_1-z_2\|^2}{\sigma^2})$$

then the following relations hold

$$\Big[\varphi(z_1)\Big]^T\varphi_x(z_2) = \nabla_{0,\,x_2}\Big[K(z_1,z_2)\Big] = \frac{2(x_1-x_2)}{\sigma^2}K(z_1,z_2),$$

$$\Big[\varphi(z_1)\Big]^T\varphi_t(z_2) = \nabla_{0,\,t_2}\Big[K(z_1,z_2)\Big] = \frac{2(t_1-t_2)}{\sigma^2}K(z_1,z_2).$$

3.5 Formulation of the Method

The general form of a linear second-order PDE with two independent variables x and t is

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial t} + c\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial x} + e\frac{\partial u}{\partial t} + l_1 u = l_2. \tag{3.27}$$

The first three terms containing the second derivatives are called the principal part of the PDE. The coefficients of the principal part can be used to classify the PDE into elliptic, parabolic and hyperbolic. In the case that the coefficients a, b and c are variable (i.e. functions of x or y, or both), then the categorization of the equation could vary throughout the solution region. Consider the one space dimensional linear second order equation with variable coefficients of the following form

$$\mathcal{L}u(z) = f(z), \quad z \in \Sigma \in \mathbb{R}^2 \tag{3.28}$$

subject to the boundary conditions of the form

$$\mathcal{B}u(z) = g(z), \quad z \in \partial\Sigma$$

where $u(z) = u(x,t)$, t and x are time and space variables respectively and $z = (x,t)^T$. Σ is a bounded domain, which can be either rectangular or irregular, and $\partial\Sigma$ represents its boundary. $\mathcal{B}$ and $\mathcal{L}$ are differential operators. In this study we consider the case where $\mathcal{L}$ is defined as follows

$$\mathcal{L} \equiv \frac{\partial^2 u}{\partial t^2} + a(x,t)\frac{\partial u}{\partial t} + b(x,t)u - c(x,t)\frac{\partial^2 u}{\partial x^2}. \tag{3.29}$$

Remark 3.5.1. *It should be noted that the presented approach can be applied for the general linear second order PDE (3.27) but for the sake of notational simplicity, the method is given for the differential operator $\mathcal{L}$ in (3.29).*

Let us assume that a general approximate solution to (3.28) is of the form of $\hat{u}(z) = w^T\varphi(z) + d$, where w and d are parameters of the model that have to be determined. To obtain the optimal value of these parameters, collocation methods can be used which assume a discretization of the domain Σ into a set of collocation points defined as follows

$$\mathcal{Z} = \left\{ z^k \,\middle|\, z^k = (x_k, t_k),\, k = 1, \ldots, M \right\},$$

where M is a user defined number. Let us decompose $\mathcal{Z}$ into two disjoint non-empty sets $\mathcal{Z}_\mathcal{D}$ and $\mathcal{Z}_\mathcal{B}$, i.e. $\mathcal{Z} = \mathcal{Z}_\mathcal{D} \cup \mathcal{X}_\mathcal{B}$, where $\mathcal{Z}_\mathcal{D} = \{z_\mathcal{D}^i\}_{i=1}^{|\mathcal{Z}_\mathcal{D}|}$ and $\mathcal{Z}_\mathcal{B} = \{z_\mathcal{B}^i\}_{i=1}^{|\mathcal{Z}_\mathcal{B}|}$. Here $|\mathcal{Z}_\mathcal{D}|$ and $|\mathcal{Z}_\mathcal{B}|$ are the cardinality of sets $\mathcal{Z}_\mathcal{D}$ and $\mathcal{Z}_\mathcal{B}$ respectively. $\mathcal{Z}_\mathcal{D}$ denotes the set of collocation points located inside the domain and $\mathcal{Z}_\mathcal{B}$ represents the collocation points situated on the boundary. Therefore the adjustable parameters w and d are to be found by solving the following optimization problem:

$$\begin{aligned} \underset{\hat{u}}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=1}^{|\mathcal{Z}_\mathcal{D}|}\left[(\mathcal{L}[\hat{u}] - f)(z_\mathcal{D}^i)\right]^2 \\ \text{subject to} \quad & \mathcal{B}[\hat{u}(z_\mathcal{B}^j)] = g(z_\mathcal{B}^j), \quad j = 1, \ldots, |\mathcal{Z}_\mathcal{B}|. \end{aligned} \tag{3.30}$$

Here $|\mathcal{Z}_\mathcal{D}|+|\mathcal{Z}_\mathcal{B}|$ is equal to the number of training points used in the learning process (see Fig. 3.3).

In what follows we formulate the optimization problem in the LSSVM framework for solving the linear second order time varying partial differential equation given in (3.28), (3.29). Suppose that $z_i \in \mathcal{S}$ and $z_j \in \mathcal{T}$ are two arbitrary points and $\mathcal{S}$, $\mathcal{T} \subseteq \mathbb{R}^2$. Now for notational convenience let us list the following notations which are used in the following sections:

$$\left[\Omega_{s^{(n)},\,p^{(m)}}\right]_{i,j}^{\mathcal{S},\mathcal{T}} = \left[\nabla_{s^{(n)},\,p^{(m)}} K\right](z_i, z_j),$$

$$\left[\Omega\right]_{i,j}^{\mathcal{S},\mathcal{T}} = \left[\nabla_{s^{(0)},\,p^{(0)}} K\right](z_i, z_j) = \left[\nabla_{0,0} K\right](z_i, z_j) = K(z_i, z_j),$$

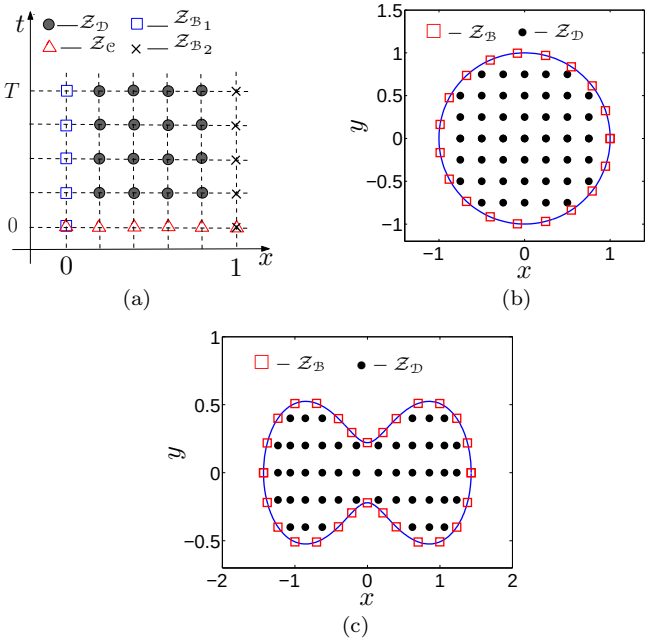


Figure 3.3: $\mathcal{Z}_{\mathcal{D}}$ and $\mathcal{Z}_{\mathcal{B}}$ are the sets of grid points which are located inside and on the boundary of the domain respectively. (a) Grid points used in the learning process for the rectangular domain, (b) Grid points used in the learning process for the circular domain, (c) Grid points used in the learning process for the irregular domain.

where $\left[\Omega_{s^{(n)},p^{(m)}}\right]_{i,j}^{\mathcal{S},\mathcal{T}}$ denotes the (i,j)-th entry of matrix $\left[\Omega_{s^{(n)},p^{(m)}}\right]^{\mathcal{S},\mathcal{T}}$. In the case that $\mathcal{S}=\mathcal{T}$, we denote the matrix by $\left[\Omega_{s^{(n)},p^{(m)}}\right]^{\mathcal{S}}$. Here s and z can take values for any t_1, t_2, x_1 and x_2 combinations see (3.4).

3.5.1 PDEs on rectangular domains

Consider the PDE (3.28), with the operator $\mathcal{L}$ in (3.29), defined on a rectangular domain subject to the initial conditions of the form

$$u(x,0)=h_0(x),\quad \frac{\partial u(x,0)}{\partial t}=h_1(x),\quad 0\le x\le 1 \tag{3.31}$$

and boundary conditions at $x = 0$ and $x = 1$ of the form

$$u(0,t) = g_0(t), \;\; u(1,t) = g_1(x), \;\; 0 \le t \le T. \tag{3.32}$$

Therefore now the set $\mathcal{Z}_{\mathcal{B}}$, defined previously, can be written as $\mathcal{Z}_{\mathcal{B}} = \mathcal{Z}_{\mathcal{C}} \cup \mathcal{Z}_{\mathcal{B}_1} \cup \mathcal{Z}_{\mathcal{B}_2}$, (see Fig 1), where

$$\mathcal{Z}_{\mathcal{C}} = \Big\{ (x,0) \,\Big|\, \forall x \in [0,1] \Big\},$$

$$\mathcal{Z}_{\mathcal{B}_1} = \Big\{ (0,t) \,\Big|\, \forall t \in [0,T] \Big\},$$

$$\mathcal{Z}_{\mathcal{B}_2} = \Big\{ (1,t) \,\Big|\, \forall t \in [0,T] \Big\}.$$

Furthermore let us assume that $N = |z_{\mathcal{D}}|$, $M_1 = |\mathcal{Z}_{\mathcal{C}}|$, $M_2 = |\mathcal{Z}_{\mathcal{B}_1}|$ and $M_3 = |\mathcal{Z}_{\mathcal{B}_2}|$.

In the LSSVM framework the approximate solution, $\hat{u}(z) = w^T\varphi(z) + d$, can be obtained by solving the following optimization problem:

$$\begin{aligned}
\underset{w,d,e}{\text{minimize}} \quad & \frac{1}{2}w^T w + \frac{\gamma}{2}e^T e \\
\text{subject to} \quad & w^T \Big[\varphi_{tt}(z^i_{\mathcal{D}}) + a(z^i_{\mathcal{D}})\varphi_t(z^i_{\mathcal{D}}) + b(z^i_{\mathcal{D}})\varphi(z^i_{\mathcal{D}}) - \\
& \qquad c(z^i_{\mathcal{D}})\varphi_{xx}(z^i_{\mathcal{D}}) \Big] + b(z^i_{\mathcal{D}})d = f(z^i_{\mathcal{D}}) + e_i, \; i = 1, \ldots, |\mathcal{Z}_{\mathcal{D}}|, \\
& w^T\varphi(z^i_{\mathcal{C}}) + d = h_0(x_i), \; i = 1, \ldots, |\mathcal{Z}_{\mathcal{C}}|, \\
& w^T\varphi_t(z^i_{\mathcal{C}}) = h_1(x_i), \; i = 1, \ldots, |\mathcal{Z}_{\mathcal{C}}|, \\
& w^T\varphi(z^i_{\mathcal{B}_1}) + d = g_0(t_i), \; i = 1, \ldots, |\mathcal{Z}_{\mathcal{B}_1}|, \\
& w^T\varphi(z^i_{\mathcal{B}_2}) + d = g_1(t_i), \; i = 1, \ldots, |\mathcal{Z}_{\mathcal{B}_2}|,
\end{aligned} \tag{3.33}$$

where

$$\varphi_t = \frac{\partial\varphi}{\partial t}, \; \varphi_{tt} = \frac{\partial^2\varphi}{\partial t^2}, \; \varphi_{xx} = \frac{\partial^2\varphi}{\partial x^2}, \; \varphi_x = \frac{\partial\varphi}{\partial x}.$$

Problem (3.33) is obtained by combining the LSSVM cost function with constraints constructed by imposing the approximate solution $\hat{u}(z) = w^T\varphi(z) +$

d, given by the LSSVM model, to satisfy the given differential equation as well as the initial and boundary conditions at the collocation points. We note here that problem (3.33) is a quadratic minimization under linear equality constraints, which enables an efficient solution.

Lemma 3.5.1. *Given a positive definite kernel function $K : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ with $K(z_1, z_2) = \varphi(z_1)^T \varphi(z_2)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (3.33) is given by the following dual problem [118]:*

$$\left[\begin{array}{c|c|c} \mathcal{K} + \gamma^{-1} I_N & S & b \\ \hline S^T & \Delta & P \\ \hline b^T & P^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \\ \hline d \end{array}\right] = \left[\begin{array}{c} f \\ \hline v \\ \hline 0 \end{array}\right]. \tag{3.34}$$

The elements of (3.34) are given by:

$$\alpha = [\alpha_1, \ldots, \alpha_N]^T,\ \beta = [\beta^1, \beta^2, \beta^3, \beta^4]^T,\ \beta^1 = [\beta^1_1, \ldots, \beta^1_{M_1}],$$

$$\beta^2 = [\beta^2_1, \ldots, \beta^2_{M_1}],\ \beta^3 = [\beta^3_1, \ldots, \beta^3_{M_2}],\ \beta^4 = [\beta^4_1, \ldots, \beta^4_{M_3}],$$

$$v = vec[H_0, H_1, G_0, G_1] \in \mathbb{R}^{2M_1+M_2+M_3},\ H_0 = [h_0(z^1_{\mathcal{C}}), \ldots, h_0(z^{M_1}_{\mathcal{C}})]^T \in \mathbb{R}^{M_1},$$

$$H_1 = [h_1(z^1_{\mathcal{C}}), \ldots, h_1(z^{M_1}_{\mathcal{C}})]^T \in \mathbb{R}^{M_1},\quad G_0 = [g_0(z^1_{\mathcal{B}_1}), \ldots, g_0(z^{M_2}_{\mathcal{B}_1})]^T \in \mathbb{R}^{M_2},$$

$$G_1 = [g_1(z^1_{\mathcal{B}_2}), \ldots, g_1(z^{M_3}_{\mathcal{B}_2})]^T \in \mathbb{R}^{M_3},\ \Delta = \left[\begin{array}{cccc} \Delta_{11} & \Delta_{12} & \Delta_{13} & \Delta_{14} \\ \Delta_{12}^T & \Delta_{22} & \Delta_{23} & \Delta_{24} \\ \Delta_{13}^T & \Delta_{23}^T & \Delta_{33} & \Delta_{34} \\ \Delta_{14}^T & \Delta_{24}^T & \Delta_{34}^T & \Delta_{44} \end{array}\right],$$

$$\Delta_{11} = \Big[\Omega\Big]^{\mathcal{X}_{\mathcal{C}}}, \Delta_{12} = \Big[\Omega_{t_1,0}\Big]^{\mathcal{Z}_{\mathcal{C}}}, \Delta_{13} = \Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{C}}},\quad \Delta_{14} = \Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{C}}}$$

$$\Delta_{21} = \Big[\Omega_{0,t_2}\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{C}}}, \Delta_{22} = \Big[\Omega_{t_1,t_2}\Big]^{\mathcal{Z}_{\mathcal{C}}}, \Delta_{23} = \Big[\Omega_{0,t_2}\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{C}}},$$

$$\Delta_{33} = \Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{B}_1}},\ \Delta_{34} = \Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{B}_1}}, \Delta_{44} = \Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{B}_2}},$$

$$S = [S_{\mathcal{C}}, \hat{S}_{\mathcal{C}}, S_{\mathcal{B}_1}, S_{\mathcal{B}_2}],\quad P = [1_{M_1}, 0_{M_1}, 1_{M_2}, 1_{M_2}],$$

$$S_{\mathcal{C}} = \Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}} + D_a\Big[\Omega_{0,t_2}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}} + D_b\Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}} - D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}}$$

$$\hat{S}_{\mathcal{C}} = \Big[\Omega_{t_1,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}} + D_a\Big[\Omega_{t_1,t_2}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}} + D_b\Big[\Omega_{t_1,0}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}} - D_c\Big[\Omega_{t_1,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{C}},\mathcal{Z}_{\mathcal{D}}},$$

$$S_{\mathcal{B}_1} = \Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}} + D_a\Big[\Omega_{0,t_2}\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}} +$$

$$D_b\Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}} - D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_1},\mathcal{Z}_{\mathcal{D}}},$$

$$S_{\mathcal{B}_2} = \Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}} + D_a\Big[\Omega_{0,t_2}\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}} +$$

$$D_b\Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}} - D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{B}_2},\mathcal{Z}_{\mathcal{D}}},$$

$$\mathcal{K} = \Big[\Omega_{t_1^{(2)},t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}} + D_a\Big[\Omega_{t_1,t_2}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_a + D_b\Big[\Omega\Big]^{\mathcal{Z}_{\mathcal{D}}}D_b +$$

$$D_c\Big[\Omega_{x_1^{(2)},x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c + \Big(D_a\Big[\Omega_{t_1^{(2)},t_2}\Big]^{\mathcal{Z}_{\mathcal{D}}} + \Big[\Omega_{t_1,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_a\Big)$$

$$+ \Big(D_b\Big[\Omega_{t_1^{(2)},0}\Big]^{\mathcal{Z}_{\mathcal{D}}} + \Big[\Omega_{0,t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_b\Big) -$$

$$\Big(D_c\Big[\Omega_{t_1^{(2)},x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}} + \Big[\Omega_{x_1^{(2)},t_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c\Big) +$$

$$\Big(D_b\Big[\Omega_{t_1,0}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_a + D_a\Big[\Omega_{0,t_2}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_b\Big)$$

$$- \Big(D_c\Big[\Omega_{t_1,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_a + D_a\Big[\Omega_{x_1^{(2)},t_2}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c\Big) -$$

$$\Big(D_c\Big[\Omega_{0,x_2^{(2)}}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_b + D_b\Big[\Omega_{x_1^{(2)},0}\Big]^{\mathcal{Z}_{\mathcal{D}}}D_c\Big) \in \mathbb{R}^{N\times N},$$

$$D_a = diag\Big(a(z_{\mathcal{D}}^1),\ldots,a(z_{\mathcal{D}}^N)\Big),$$

$$D_b = diag\Big(b(z_{\mathcal{D}}^1),\ldots,b(z_{\mathcal{D}}^N)\Big),\ f = [f(z_{\mathcal{D}}^1),\ldots,f(z_{\mathcal{D}}^N)]^T$$

$$D_c = diag\Big(c(z_{\mathcal{D}}^1),\ldots,c(z_{\mathcal{D}}^N)\Big),\ b = [b(z_{\mathcal{D}}^1),\ldots,b(z_{\mathcal{D}}^N)]^T,$$

where $vec(\cdot)$ denotes the vectorization of a matrix. Also note that $\mathcal{K} = \mathcal{K}^T$.

The dual model representation of the solution is as follows:

$$\begin{aligned}\hat{u}(z) =& d + \sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|} \alpha_i \Bigg(\Big[\nabla_{t_1^{(2)},0}\, K\Big](z_{\mathcal{D}}^i, z) + a(z_{\mathcal{D}}^i)\Big[\nabla_{t_1,0}\, K\Big](z_{\mathcal{D}}^i, z) + \\ & b(z_{\mathcal{D}}^i)\Big[\nabla_{0,0}\, K\Big](z_{\mathcal{D}}^i, z) - c(z_{\mathcal{D}}^i)\Big[\nabla_{x_1^{(2)},0}\, K\Big](z_{\mathcal{D}}^i, z)\Bigg) + \\ & \sum_{i=1}^{|\mathcal{Z}_{\mathcal{C}}|} \beta_i^1 \Big[\nabla_{0,0}\, K\Big](z_{\mathcal{C}}^i, z) + \sum_{i=1}^{|\mathcal{Z}_{\mathcal{C}}|} \beta_i^2 \Big[\nabla_{t_1,0}\, K\Big](z_{\mathcal{C}}^i, z) + \\ & \sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_1}|} \beta_i^3 \Big[\nabla_{0,0}\, K\Big](z_{\mathcal{B}_1}^i, z) + \sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}_2}|} \beta_i^4 \Big[\nabla_{0,0}\, K\Big](z_{\mathcal{B}_2}^i, z).\end{aligned}$$

3.5.2 PDEs on irregular domains

Consider the PDE (3.28), with operator $\mathcal{L}$ in (3.29), defined on a irregular domain subject to a Dirichlet boundary condition, i.e.

$$u(z) = g(z) \;\text{ for all }\; z \in \partial\Sigma.$$

The approximate solution, $\hat{u}(z) = w^T\varphi(z) + d$, can then be obtained by solving the following optimization problem,

$$\begin{aligned}\underset{w,d,e}{\text{minimize}} \quad & \frac{1}{2}w^T w + \frac{\gamma}{2}e^T e \\ \text{subject to} \quad & w^T\Big[\varphi_{tt}(z_{\mathcal{D}}^i) + a(z_{\mathcal{D}}^i)\varphi_t(z_{\mathcal{D}}^i) + b(z_{\mathcal{D}}^i)\varphi(z_{\mathcal{D}}^i) - \\ & \qquad c(z_{\mathcal{D}}^i)\varphi_{xx}(z_{\mathcal{D}}^i)\Big] + b(z_{\mathcal{D}}^i)d = \\ & \qquad f(z_{\mathcal{D}}^i) + e_i,\; i = 1, \ldots, |\mathcal{Z}_{\mathcal{D}}|, \\ & w^T\varphi(z_{\mathcal{B}}^i) + d = g(t_i),\; i = 1, \ldots, |\mathcal{Z}_{\mathcal{B}}|.\end{aligned} \tag{3.35}$$

Here $\mathcal{Z}_{\mathcal{D}}$ and $\mathcal{Z}_{\mathcal{B}}$ are defined as previously.

Lemma 3.5.2. *Given a positive definite kernel function $K : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ with $K(z_1, z_2) = \varphi(z_1)^T\varphi(z_2)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (3.35) is given by the following dual problem [118]:*

$$\left[\begin{array}{c|c|c} \mathcal{K}+\gamma^{-1}I_N & S_{\mathcal{B}} & b \\ \hline S_{\mathcal{B}}^T & \Delta_{\mathcal{B}} & 1_M \\ \hline b^T & 1_M^T & 0 \end{array}\right]\left[\begin{array}{c} \alpha \\ \hline \beta \\ \hline d \end{array}\right]=\left[\begin{array}{c} f \\ \hline g \\ \hline 0 \end{array}\right] \tag{3.36}$$

with

$$N=|\mathcal{Z}_{\mathcal{D}}|,\ M=|\mathcal{Z}_{\mathcal{B}}|, \beta=[\beta_1,\ldots,\beta_M]^T\in\mathbb{R}^M,$$

$$g=[g(z_{\mathcal{B}}^1),\ldots,g(z_{\mathcal{B}}^M)]^T\in\mathbb{R}^M, \Delta_{\mathcal{B}}=\left[\Omega\right]^{\mathcal{Z}_{\mathcal{B}}}\in\mathbb{R}^{M\times M},$$

$$S_{\mathcal{B}}=\left[\Omega_{0,t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}+D_a\left[\Omega_{0,t_2}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}+$$

$$D_b\left[\Omega\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}-D_c\left[\Omega_{0,x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}}$$

where $\mathcal{K}, b, f, D_a, D_b, \alpha$ and D_c are defined as previously.

The dual model representation of the solution is as follows:

$$\hat{u}(z)=\sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|}\alpha_i\left(\left[\nabla_{t_1^{(2)},0}K\right](z_{\mathcal{D}}^i,z)+a(z_{\mathcal{D}}^i)\left[\nabla_{t_1,0}K\right](z_{\mathcal{D}}^i,z)+b(z_{\mathcal{D}}^i)\left[\nabla_{0,0}K\right](z_{\mathcal{D}}^i,z)\right.$$

$$\left.-c(z_{\mathcal{D}}^i)\left[\nabla_{x_1^{(2)},0}K\right](z_{\mathcal{D}}^i,z)\right)+\sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}}|}\beta_i\left[\nabla_{0,0}K\right](z_{\mathcal{B}}^i,z)+d.$$

Proof. It follows from constructing the Lagrangian of the constrained optimization (3.35) as in Lemma 2.1, then obtaining the Karush-Kuhn-Tucker optimality condition and eliminating the primal variables w and e. □

The LSSVM model for the solution derivative, with respect to space (x) and time (t), in the dual form become:

$$\frac{\partial \hat{u}(z)}{\partial x} = \sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|} \alpha_i \left(\left[\nabla_{t_1^{(2)},x_2} K \right](z_{\mathcal{D}}^i, z) + a(z_{\mathcal{D}}^i) \left[\nabla_{t_1,x_2} K \right](z_{\mathcal{D}}^i, z) + \right.$$

$$\left. b(z_{\mathcal{D}}^i) \left[\nabla_{0,x_2} K \right](z_{\mathcal{D}}^i, z) - c(z_{\mathcal{D}}^i) \left[\nabla_{x_1^{(2)},x_2} K \right](z_{\mathcal{D}}^i, z) \right) + \sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}}|} \beta_i \left[\nabla_{0,x_2} K \right](z_{\mathcal{B}}^i, z),$$

$$\frac{\partial \hat{u}(z)}{\partial t} = \sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|} \alpha_i \left(\left[\nabla_{t_1^{(2)},t_2} K \right](z_{\mathcal{D}}^i, z) + a(z_{\mathcal{D}}^i) \left[\nabla_{t_1,t_2} K \right](z_{\mathcal{D}}^i, z) + b(z_{\mathcal{D}}^i) \left[\nabla_{0,t_2} K \right](z_{\mathcal{D}}^i, z) \right.$$

$$\left. - c(z_{\mathcal{D}}^i) \left[\nabla_{x_1^{(2)},t_2} K \right](z_{\mathcal{D}}^i, z) \right) + \sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}}|} \beta_i \left[\nabla_{0,t_2} K \right](z_{\mathcal{B}}^i, z),$$

where K is the kernel function.

Remark 3.5.2. *Although in section 3.5.2, the formulation of the method is presented for a Dirichlet boundary condition, it can be adapted, by adopting suitable constraints, for the Neumann or Robin (a linear combination of the Dirichlet and Neumann) type boundary conditions. Furthermore, the formulation can also be applied for a rectangular domain by incorporating suitable set of constraints satisfying the initial/boundary conditions.*

3.5.3 Formulation of the method for nonlinear PDE

Inspired by the approach described in 3.2.5 for nonlinear ODEs, we formulate an optimization problem based on least squares support vector machines for solving nonlinear partial differential equations. For the sake of notational simplicity let us assume the the nonlinear PDE has the following form:

$$\frac{\partial^2 u}{\partial t^2} + \frac{\partial^2 u}{\partial x^2} + f(u) = g(z), \quad z \in \Sigma \in \mathbb{R}^2 \tag{3.37}$$

subject to the boundary conditions of the form

$$u(z) = h(z), z \in \partial\Sigma \tag{3.38}$$

where f is a nonlinear function. The approximate solution $\hat{u}(z) = w^T \varphi(z) + d$ for the given nonlinear PDE can be obtained by solving the following optimization problem [118]:

$$
\begin{aligned}
\underset{w,d,e,\xi,u}{\text{minimize}} \quad & \frac{1}{2}w^T w + \frac{\gamma}{2}(e^T e + \xi^T \xi) \\
\text{subject to} \quad & w^T \left[\varphi_{tt}(z_{\mathcal{D}}^i) + \varphi_{xx}(z_{\mathcal{D}}^i) \right] + f(u(z_{\mathcal{D}}^i)) \\
& = g(z_{\mathcal{D}}^i) + e_i,\ i = 1, \ldots, |\mathcal{Z}_{\mathcal{D}}|, \\
& w^T \varphi(z_{\mathcal{D}}^i) + d = u(z_{\mathcal{D}}^i) + \xi_i,\ i = 1, \ldots, |\mathcal{Z}_{\mathcal{D}}|, \\
& w^T \varphi(z_{\mathcal{B}}^i) + d = h(z_{\mathcal{B}}^i),\ i = 1, \ldots, |\mathcal{Z}_{\mathcal{B}}|.
\end{aligned}
\tag{3.39}
$$

Note that the second set of additional constraints is introduced to keep the optimization problem linear in w. As before, we assume that $N = |\mathcal{Z}_{\mathcal{D}}|$, $M = |\mathcal{Z}_{\mathcal{B}}|$. After deriving the Lagrangian, taking the KKT conditions and eliminating the primal variables w, e, ξ one obtains the following nonlinear system of equations:

$$
\begin{cases}
\mathcal{K}\alpha + S_1\eta^1 + S_2\eta^2 - f(u) = 0 \\
S_1^T\alpha + \Delta_{11}\eta^1 + \Delta_{12}\eta^2 + 1_N d - I_N u = 0 \\
S_2^T\alpha + \Delta_{12}^T\eta^1 + \Delta_{22}\eta^2 + 1_M d = 0 \\
1_N^T\eta^1 + 1_M^T\eta^2 = 0 \\
diag(f_u)\alpha - \eta^1 = 0
\end{cases}
\tag{3.40}
$$

where η_1, η_2 and α are Lagrange multipliers. and $u = [u(z_{\mathcal{D}}^1), \ldots, u(z_{\mathcal{D}}^N)]^T$. $f_u = [\frac{d}{du}f(u(z_{\mathcal{D}}^1)), \ldots, \frac{d}{du}f(u(z_{\mathcal{D}}^N))]$ and $\text{diag}(f_u)$ is a diagonal matrix with elements of f_u on the diagonal.

$$
\begin{aligned}
\mathcal{K} = & \left[\Omega_{t_1^{(2)},t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}} + \left[\Omega_{x_1^{(2)},x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}} + \\
& \left[\Omega_{t_1^{(2)},x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}} + \left[\Omega_{x_1^{(2)},t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}} + \gamma^{-1} I_N \in \mathbb{R}^{N\times N} \\
S_1 = & \left[\Omega_{0,t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}} + \left[\Omega_{0,x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{D}}} \\
S_2 = & \left[\Omega_{0,t_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}} + \left[\Omega_{0,x_2^{(2)}}\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}} \\
\Delta_{11} = & \left[\Omega\right]^{\mathcal{Z}_{\mathcal{D}}} + \gamma^{-1} I_N,\ \Delta_{12} = \left[\Omega\right]^{\mathcal{Z}_{\mathcal{B}},\mathcal{Z}_{\mathcal{D}}},\ \Delta_{22} = \left[\Omega\right]^{\mathcal{Z}_{\mathcal{B}}}.
\end{aligned}
$$

The nonlinear system (3.40) is solved for $(\alpha, \eta^1, \eta^2, d, u)$ using Newton's method. The Jacobian of (3.40) can be explicitly represented as follows:

$$J = \left[\begin{array}{c|c|c|c|c} \mathcal{K} & S_1 & S_2 & 0_N & -\mathrm{diag}(f_u) \\ \hline S_1^T & \Delta_{11} & \Delta_{12} & 1_N & -I_N \\ \hline S_2^T & \Delta_{12}^T & \Delta_{22} & 1_M & 0_{M\times N} \\ \hline 0_N^T & 1_N^T & 1_M^T & 0 & 0_N^T \\ \hline \mathrm{diag}(f_u) & -I_N & 0_{N\times M} & 0_N & \mathrm{diag}(f_{uu} \odot \alpha) \end{array}\right]$$

where $f_{uu} = [\frac{d^2}{du^2} f(u(z_{\mathcal{D}}^1)), \ldots, \frac{d^2}{du^2} f(u(z_{\mathcal{D}}^N))]$ and $\odot$ denotes the element-wise multiplication. The dual model representation of the solution is as follows:

$$\hat{u}(z) = \sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|} \alpha_i \left(\left[\nabla_{t_1^{(2)},0} K\right](z_{\mathcal{D}}^i, z) + \left[\nabla_{x_1^{(2)},0} K\right](z_{\mathcal{D}}^i, z) \right) + \sum_{i=1}^{|\mathcal{Z}_{\mathcal{D}}|} \eta_i^1 \left[\nabla_{0,0} K\right](z_{\mathcal{D}}^i, z) + \sum_{i=1}^{|\mathcal{Z}_{\mathcal{B}}|} \eta_i^2 \left[\nabla_{0,0} K\right](z_{\mathcal{B}}^i, z) + d.$$

3.6 Model Selection

The performance of the LSSVM model depends on the choice of the tuning parameters. For all experiments the Gaussian RBF kernel is used. Therefore a model is determined by the regularization parameter γ and the kernel bandwidth σ. It should be noted that unlike the regression case, we do not have target values and consequently we do not have noise. Therefore a quite large value should be taken for the regularization constant γ so that the error e is sharply minimized or equivalently the constraints are well satisfied. This is also verified when a grid search over different γ values is performed. In all the experiments the chosen value for γ is approximately 10^7. Therefore the only parameter left that has to be tuned is the kernel bandwidth. In this work, the optimal values of σ are obtained by evaluating the performance of the model on a validation set using a meaningful range of possible (σ) i.e. $\{10^{-2}, \ldots, 10^3\}$. In the case of ODE and DAE problems, the validation set is defined to be the set of midpoints $V \equiv \{v_i = \frac{(t_i + t_{i+1})}{2}, i = 1, ..., N-1\}$ where $\{t_i\}_{i=1}^N$ are training points. The value of σ for which the mean squared error (MSE) on this validation set is minimum has been selected. Similar strategy has been applied for tuning the model parameters when the solution of PDEs is learned.

3.7 Experiments

In this section, we have tested the performance of the proposed method on several ODE, DAE and PDE problems. In the ODE test problems, in order to show the approximation and generalization capabilities of the proposed method, we compare the exact solution with the computed solution inside and outside of the domain of consideration. Furthermore the proposed method is successfully applied to solve problem 1 for a very large time interval. For all the experiments, the RBF kernel is used, $K(u,v) = \exp(-\frac{\|u-v\|^2}{\sigma^2})$. MATLAB 2010b is used to implement the code and all computations were carried out on a windows 7 system with Intel(R)-core(TM) i7 CPU and 4.00GB RAM. The Matlab implementation of the proposed approach can be found in `https://sites.google.com/site/smkmhr/Projects`.

3.7.1 ODE test problems

Problem 1: Consider the following first order ODE

$$\frac{d}{dt}y(t) + 2y(t) = \sin(t), \quad y(0) = 1, \quad t \in [0, 10].$$

The approximate solution obtained by the proposed method is compared with the true solution and results are depicted in Fig 3.4. In addition we also considered points outside the training interval, and Fig 3.4 (d) and (e) show that the extrapolation error remains low for the points near the domain of equation. As it was expected by increasing the number of mesh points (training points), the error decreases both inside and outside of the training interval. Fig 3.4 (c) and (f) indicate the performance of the method when non-uniform partitioning is used for creating training points. The kernel bandwidth parameter used in the simulation is $\sigma = 21.54$.

Problem 2: First order differential equation with nonlinear sinusoidal excitation

$$\frac{d}{dt}y(t) + 2y(t) = t^3 \sin(t/2), \quad y(0) = 1, \quad t \in [0, 10].$$

The interval [0,10] is discretized into $N = 20$ points $t_1 = 0, ..., t_{20} = 10$ using the grid $t_i = (i-1)h$, $i = 1, ..., N$, where $h = \frac{10}{N-1}$. In Fig 3.5(a), we compare the exact solution with the computed solution at grid points (circles) as well as for other points inside and outside the domain of equation. The obtained absolute errors for points inside and outside the domain [0,10] are tabulated in Table 3.1. The kernel bandwidth parameter used in the simulation is $\sigma = 21.40$.

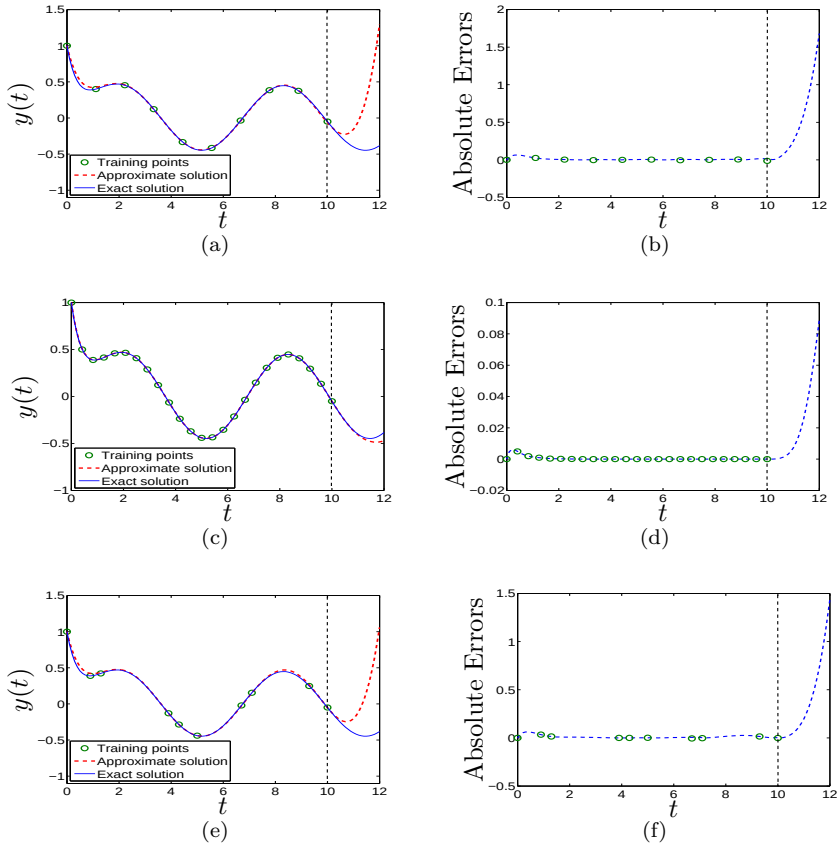


Figure 3.4: Numerical results for Problem 1. (a) 10 equidistant points in $[0, 10]$ are used for training. (c) 25 equidistant points in $[0, 10]$ are used for training. (e) Non-uniform partitions of [0,10] using 10 points which are used for training. (b) Obtained absolute errors on the interval $[0, 12]$ when $[0, 10]$ is discretized into 9 equal parts. (d) Obtained absolute errors on the interval $[0, 12]$ when $[0, 10]$ is discretized into 24 equal parts. (f) Obtained absolute errors on the interval $[0, 12]$ when $[0, 10]$ is discretized into 9 non-uniform parts.

The influence of increasing number of training points on the optimal value of the kernel bandwidth σ is also shown in Fig. 3.5(d). A comparison with MATLAB built-in solver ode45 is made. The most intuitive way of comparing the results of the two approaches is in terms of accuracy versus computational time which is shown in Fig 3.5. We analyze the scenario where the number of training points is increasing which results in improving the accuracy at the expense of increasing computation time. For the ode45 solver, the tolerance provided by the user is chosen from the set $[10^{-4}, 10^{-5}, \ldots, 10^{-9}]$. For this example, one can observe that the proposed approach requires less training computational time in order to reach a desired accuracy in range $[10^{-6}, 10^{-14}]$. This is expected as few number of training points was needed to obtain the desired accuracy. However one may notice the error saturation behavior of the proposed approach. Previous studies have also confirmed the occurrence of error saturation for many types of kernels including RBFs. (Interested readers are referred to [31, 30, 105] and references therein for more details on the saturation error for Gaussian RBFs).

Problem 3: Consider the following nonlinear first order ODE:

$$\frac{d}{dt}y(t) = y(t)^2 + t^2, \quad y(0) = 1, \quad t \in [0, 0.5].$$

Twenty equidistant points in the given interval are used for the training process. The obtained approximate solution by the proposed method and the solution obtained by MATLAB built-in solver ode45 are displayed in Fig 3.5(b). The obtained absolute errors for points inside and outside the domain [0,0.5] are tabulated in Table 3.1. The kernel bandwidth parameter used in the simulation is $\sigma = 0.2$.

Problem 4: Consider the following first order ODE with time varying coefficient [86, Problem 1]:

$$\frac{d}{dt}y(t) + \left(t + \frac{1+3t^2}{1+t+t^3}\right)y(t) = t^3 + 2t + t^2\frac{1+3t^2}{1+t+t^3},$$

$$y(0) = 1, \quad t \in [0, 1].$$

In order to have a fair comparison with the results reported in [86], ten equidistant points in the given interval are used for the training process. The analytic solution and obtained solution via our proposed method are displayed in Fig 3.5(c). The obtained absolute errors for points inside and outside the domain [0,1] are recorded in Table 3.1, which shows the superiority of the proposed method over the described method in [86]. (Note that in [86, Fig 2] the

maximum absolute error outside the domain $[0, 1]$ is approximately 12×10^{-2}). The kernel bandwidth parameter used in the simulation is $\sigma = 20.0$.

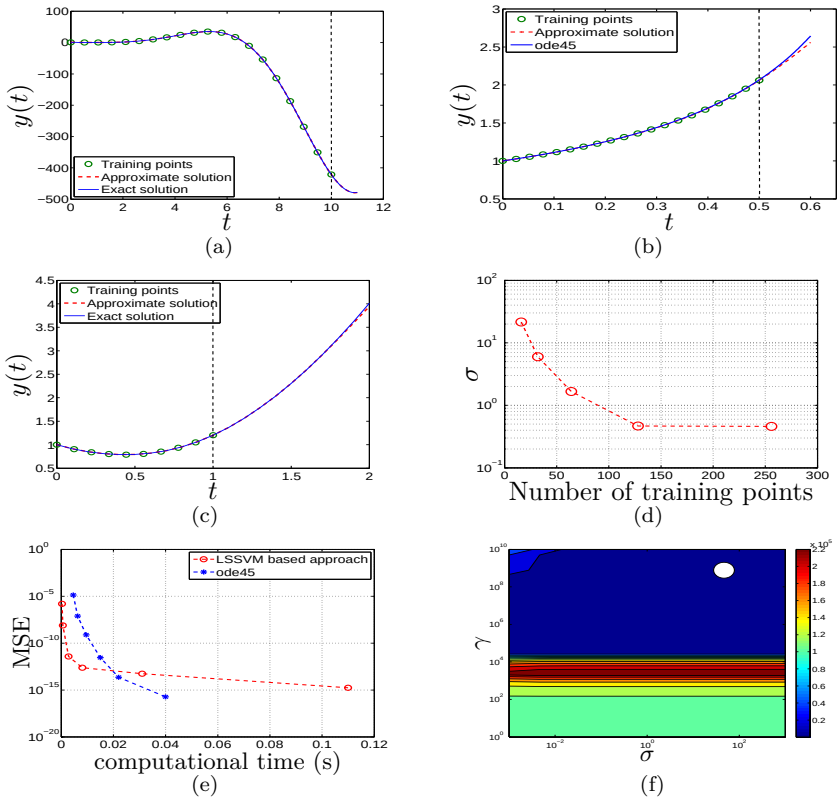


Figure 3.5: (a) Numerical results for Problem 2. Twenty equidistant points in [0,10] are used for training. (b) Numerical results for Problem 3. Twenty equidistant points in [0,0.5] are used for training. (c) Numerical results for Problem 4.4. Ten equidistant points in [0,1] are used for training. (d) Effect of increasing number of training points on the optimal σ value. (e) Comparison with ode45 solver in terms of accuracy and computational time. (f) Tuning the model parameters using the validation set.

Problem 5: Consider the following second order boundary value problem with time-varying input signal

$$\frac{d^2}{dt^2}y(t) + y(t) = 2 + 2\sin(4t)\cos(3t),$$

$$y(0) = 1, \quad y(1) = 0.$$

Ten equidistant points in the given interval are used for the training process. The analytic solution and the obtained solution via our proposed method are displayed in Fig 3.6(a). The obtained absolute errors for points inside and outside the domain [0,1] are recorded in Table 3.2. The kernel bandwidth parameter used in the simulation is $\sigma = 1.02$.

Problem 6: Consider the following second order ODE with time-varying input signal [86, Problem 3] :

$$\frac{d^2}{dt^2}y(t) + \frac{1}{5}\frac{d}{dt}y(t) + y(t) = -\frac{1}{5}e^{(-t/5)}\cos(t),$$

$$y(0) = 1, \quad y'(0) = 1.$$

Ten equidistant points in the interval [0,2] are used for the training process. The analytic solution and the obtained solution by the proposed method are shown in Fig 3.6(b). The obtained absolute errors for points inside and outside the domain [0,2] are tabulated in Table 3.2, which again shows the improvement of the proposed method over the described method in [86]. (Note that in [86, Fig 4] the maximum absolute error outside the domain $[0, 2]$ is 8×10^{-4}). The kernel bandwidth parameter used in the simulation is $\sigma = 15.63$.

Problem 7: Consider the following second order ODE [164, Example 1]:

$$\frac{d^2}{dt^2}y(t) + \frac{1}{t}\frac{d}{dt}y(t) - \frac{1}{t}\cos(t) = 0, \quad y(0) = 0, \quad y'(0) = 1.$$

$$\text{Exact solution: } \; y(t) = \int_0^t \frac{\sin(x)}{x}\, dx.$$

Ten equidistant points in the interval [0,1] are used as training points and the obtained result are shown in Fig 3.6(c) and recorded in Table 3.2. The obtained maximum absolute error outside the domain $[0, 1]$ is 6.51×10^{-2} which is smaller than 14×10^{-1} shown in [164, Fig 6]. The kernel bandwidth parameter used in the simulation is $\sigma = 2.00$.

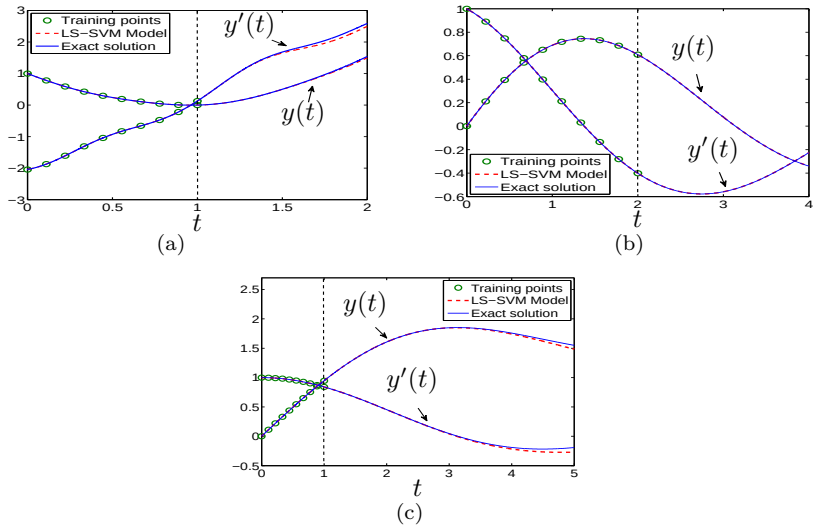


Figure 3.6: (a) Numerical results for Problem 5. Ten equidistant points in [0,1] are used for training, (b) Numerical results for Problem 6. Ten equidistant points in [0,2] are used for training. (c) Numerical results for Problem 7. Ten equidistant points in [0,1] are used for training.

Table 3.1: Numerical results of the proposed method for solving Problems 2, 3 and 4.

Problem	Domain	$\|y - \hat{y}\|_\infty$	MSE	STD
2	**Inside**	4.56×10^{-3}	1.47×10^{-6}	1.16×10^{-3}
	Outside	4.62×10^{-1}	3.85×10^{-2}	1.56×10^{-1}
3	**Inside**	5.43×10^{-3}	8.94×10^{-6}	1.60×10^{-3}
	Outside	8.46×10^{-2}	1.49×10^{-3}	2.27×10^{-2}
4	**Inside**	1.46×10^{-4}	8.15×10^{-9}	3.90×10^{-5}
	Outside	6.76×10^{-2}	5.53×10^{-4}	2.20×10^{-2}

Note: MSE is the mean squared error and STD is the stand deviation.

3.7.1.1 Sensitivity of the solution w.r.t the parameter

In order to illustrate the sensitivity of the result with respect to the parameter of the model (σ), for two examples we have plotted the MSE, on the validation set, versus the kernel bandwidth on logarithmic scales in Fig 3.7. From this figure, it is apparent that there exists a range of σ for which the MSE on the validation set is quite small.

Table 3.2: Numerical results of the proposed method for solving Problems 5, 6 and 7.

Problem	Domain	Variable	$\|\boldsymbol{y}-\hat{\boldsymbol{y}}\|_\infty$	MSE	STD
5	**Inside**	y	1.14×10^{-6}	4.16×10^{-13}	6.43×10^{-7}
		y'	4.81×10^{-5}	6.78×10^{-11}	8.21×10^{-6}
	Outside	y	4.20×10^{-2}	2.64×10^{-4}	1.26×10^{-2}
		y'	1.00×10^{-1}	3.27×10^{-3}	3.87×10^{-2}
6	**Inside**	y	5.88×10^{-6}	1.49×10^{-11}	1.63×10^{-6}
		y'	7.34×10^{-6}	2.18×10^{-11}	3.28×10^{-6}
	Outside	y	3.96×10^{-4}	2.39×10^{-8}	1.19×10^{-4}
		y'	5.15×10^{-4}	7.11×10^{-8}	1.74×10^{-4}
7	**Inside**	y	6.64×10^{-9}	2.01×10^{-16}	4.07×10^{-9}
		y'	8.41×10^{-8}	1.30×10^{-15}	3.59×10^{-8}
	Outside	y	6.51×10^{-2}	3.90×10^{-4}	1.65×10^{-2}
		y'	7.80×10^{-2}	7.31×10^{-4}	2.15×10^{-2}

Note: MSE is the mean squared error and STD is the stand deviation.

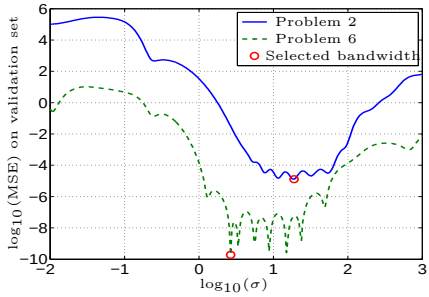


Figure 3.7: Sensitivity of the obtained result with respect to model parameter σ. $\log_{10}$(MSE) vs. $\log_{10}\sigma$ is plotted for Problems 2 and 6.

3.7.1.2 Large interval

Let us consider problem 1 when the time interval is $[0, 10^5]$. It is known in advance that the solution of this problem is oscillating. The problem is solved by decomposing the given interval of interest into S sub-intervals. Then the problem is solved on each sub-interval using N local collocation points. The execution time and the mean squared error (MSE) for the training and test sets

$$MSE_{train} = \frac{\sum_{i=1}^{N\times S}(y(t_i)-\hat{y}(t_i))^2}{N\times S},$$

$$MSE_{test} = \frac{\sum_{i=1}^{M}(y(t_i)-\hat{y}(t_i))^2}{M},$$

where $N \times S$ is the total number of collocation points and M is the total number of test points over the interval $[0, 10^5]$, are tabulated in Table 3.3. The test set is the same for all the cases and it consists of $M = 5 \times 10^5$ points. It is apparent that when S is fixed and N increases, the accuracy is improved whereas the execution time is increased. The same pattern is observed when N is fixed and S increases. Fig. 3.8(a) and (b) show the residual error $e_t = y(t) - \hat{y}(t)$ when Problem 1 is solved over the interval $[0, 10^5]$, using $N = 50$ local collocation points, $S = 5000$ sub-intervals and $N = 500$, collocation points, $S = 500$ sub-intervals respectively. It should be noted that the result depicted in Fig. 3.8(a) is obtained much faster than that shown in Fig. 3.8(b).

In Table 3.4, we analyze the situation where the total number of collocation points i.e. $N \times S$ in the given interval $[0, 4000]$ is fixed. It can be seen that as the number of sub-intervals increases (number of collocation points in each sub-interval increases) the computational time decreases without losing the order of accuracy. In this case the test set consists of $M = 2 \times 10^4$ points.

Table 3.3: Numerical result of the proposed method for solving Problem 1 with time interval $[0, 10^5]$. N is the number of local collocation points and S is the number sub-intervals.

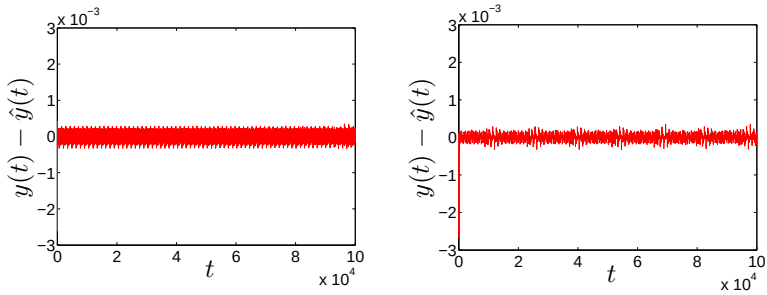
			MSE	
N	S	**CPU time**	Training	Test
20	1000	5.5	2.4×10^{-2}	7.2×10^{-2}
	2000	10.6	1.3×10^{-3}	3.3×10^{-3}
	5000	29.5	8.4×10^{-8}	2.3×10^{-7}
30	1000	6.6	2.2×10^{-2}	5.9×10^{-2}
	2000	13.4	4.1×10^{-6}	1.3×10^{-5}
	5000	37.1	8.2×10^{-9}	2.7×10^{-8}
40	1000	9.6	5.8×10^{-4}	1.4×10^{-3}
	2000	20.1	1.7×10^{-7}	5.8×10^{-7}
	5000	54.2	2.3×10^{-9}	8.1×10^{-9}

Note: The execution time is in seconds.

Table 3.4: Numerical results of the proposed method for solving Problem 1 with time interval [0,4000], while total number of collocation points i.e. $N \times S$ is constant

			MSE	
N	S	**CPU time**	Training	Test
800	10	85.5	1.36×10^{-8}	2.06×10^{-8}
400	20	26.1	1.37×10^{-8}	2.08×10^{-8}
20	400	2.06	1.68×10^{-8}	2.52×10^{-8}

Note: The execution time is in seconds.



(a) (b)

Figure 3.8: (a) Residual $y(t) - \hat{y}(t)$ when problem 1 is solved on the interval $[0, 10^5]$, by using 5000 sub-intervals and 50 local collocation points. (b) Obtained residual $y(t) - \hat{y}(t)$ for the same problem by using 500 sub-intervals and 500 local collocation points.

3.7.2 DAE test problems

Three experiments are performed to demonstrate the capability of the proposed method for solving initial and boundary value problems in DAEs. The accuracy of an approximate solution is measured by means of mean squared error (MSE) which is defined as follows:

$$\text{MSE}_{\text{test}} = \frac{\sum_{i=1}^{M}(x(t_i) - \hat{x}(t_i))^2}{M}$$

where M is the number test points. In all the experiments M is set to 200 points on the given domain.

Problem 8: Consider the following nonsingular system of time varying ordinary differential equations

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-2}{t^2} & \frac{2}{t^2} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ t\log(t) \end{bmatrix} \tag{3.41}$$

subject to

$$x_1(1) = 1, x_2(1) = 0.$$

This problem is solved for $t \in [1, 10]$ and the approximate solution obtained by the proposed method is compared with the solution obtained by MATLAB built-in solver ode45 in Fig 3.9. The obtained results with different numbers of training points are tabulated in Table 3.5. Note that the subroutine DSolve of MATHEMATICA 6.0 failed to find the analytical solution for the above equation. The kernel bandwidth parameter used in the simulation is $\sigma = 1.67$.

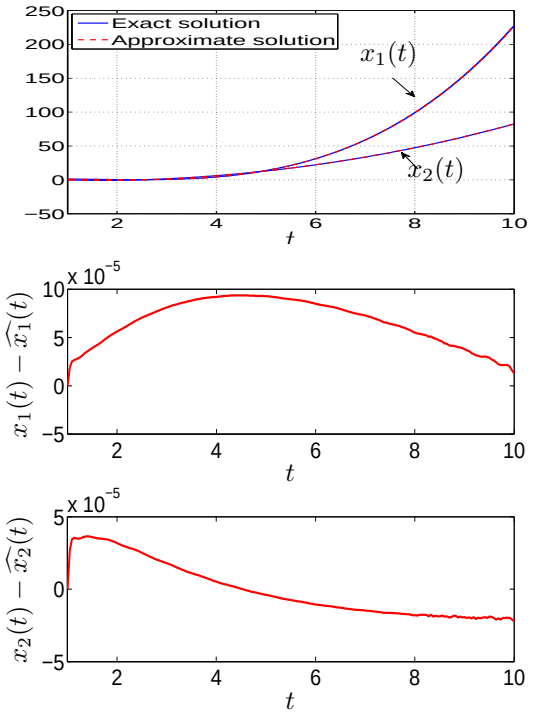


Figure 3.9: Obtained approximate solution and model errors for problem 8, when 80 equidistant mesh points on the interval [1,10] are used for training.

Table 3.5: Numerical results of the proposed method for solving Problem 8 on time interval [1,10], with N number of collocation points.

N	$\mathbf{MSE}_{\text{test}}$	
	x_1	x_2
20	3.1×10^{-2}	1.2×10^{-3}
40	3.9×10^{-5}	1.4×10^{-6}
60	2.6×10^{-7}	1.1×10^{-8}
80	4.8×10^{-9}	3.5×10^{-10}

Problem 9: Consider the singular system of index-3 discussed as follows [122]:

$$Z(t)\dot{X}(t) = A(t)X(t) + B(t)u(t),\ t \in [0, 20],\ X(0) = X_0 \tag{3.42}$$

where

$$Z = \begin{bmatrix} 0 & -t & 0 \\ 1 & 0 & t \\ 0 & 1 & 0 \end{bmatrix},\ A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

and $B(t) = 0$ with $x(0) = [0, e^{-1}, e^{-1}]^T$. The exact solution is given by

$$x_1(t) = -t\exp(-(t+1)),\ \ x_2(t) = x_3(t) = \exp(-(t+1)).$$

The problem is solved on domain $t \in [0, 20]$ for different N (number of collocation points) values. The approximate solution obtained by the proposed method is compared with the exact solution (see Fig 3.10) and the results are recorded in Table 3.6. From Table 3.6, it is apparent that as N increases, the solution converges to the true solution. Note that the MATLAB built-in solver ode15i can solve DAEs Up to index-1. The kernel bandwidth parameter used in the simulation is $\sigma = 4.49$.

Table 3.6: Numerical results of the proposed method for solving Problem 9 on time interval [0,20], with N number of collocation points.

	$\mathbf{MSE}_{\text{test}}$		
N	x_1	x_2	x_3
20	1.33×10^{-5}	4.82×10^{-8}	4.73×10^{-7}
40	1.38×10^{-8}	1.39×10^{-10}	3.14×10^{-9}
60	4.82×10^{-10}	3.54×10^{-12}	2.38×10^{-10}

Problem 10: Consider the linear time varying singular system [123]

$$\begin{aligned} Z(t)\dot{X}(t) &= A(t)X(t) + B(t)u(t),\ t \in [0, 10],\ X(0) = X_0 \\ y(t) &= C(t)x(t) \end{aligned} \tag{3.43}$$

where $y(t)$ is the output vector and

$$Z = \begin{bmatrix} 1+t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1+t & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -(1+t) & 1 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 0 & 1+t & 0 & -1 \\ 0 & 0 & 0 & (1+e^{-t})\sin(t) \end{bmatrix}$$

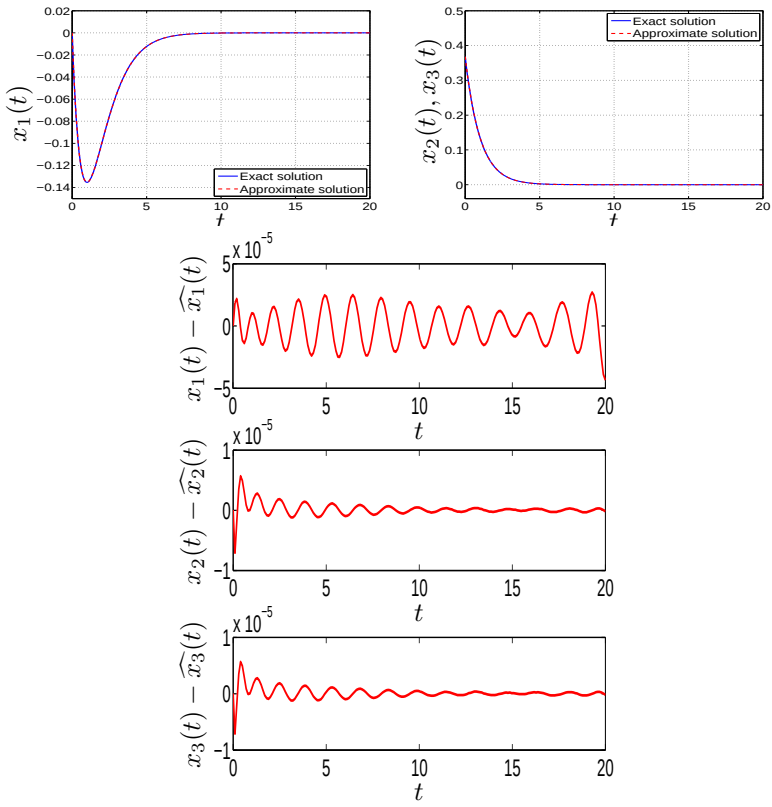


Figure 3.10: Obtained approximate solution and model errors for problem 9, when 70 equidistant mesh points on the interval [0,20] are used for training.

with $u = [1+t+t^2/2, 0]^T$ and $x(0) = [0, -1, 0, -1]^T$ the exact solution is given by

$$y_1(t) = -\left(1 + t + t^2 + \frac{t^3}{3}\right) + \frac{1 + t + 3t^2/2 + t^3 + t^4/4}{1 + t},$$

$$y_2(t) = -(1 + e^{-t}) \sin(t) \left(\frac{1 + t + 3t^2/2 + t^3 + t^4/4}{1 + t}\right).$$

The interval [0,10] is discretized into N points. The obtained mean squared errors for test set are tabulated in Table 3.7. The results reveal that higher order accuracy can be achieved by an increasing number of collocation points. Analytical solution and obtained approximate solution, with 50 equidistant mesh points on the interval $[0, 10]$ as training points, are compared in Fig 3.11.

Note that the MATLAB built-in solver ode15i failed to solve problem 10. The kernel bandwidth parameter used in the simulation is $\sigma = 10.00$.

Table 3.7: Numerical results of the proposed method for solving Problem 10 on time interval [0,10], with N collocation points.

N	$\mathbf{MSE}_{\text{test}}$	
	y_1	y_2
10	1.23×10^{-3}	8.76×10^{-4}
20	1.25×10^{-5}	1.31×10^{-5}
30	1.42×10^{-6}	7.80×10^{-7}
40	5.35×10^{-9}	7.62×10^{-9}

3.7.3 PDE test problems

Problem 11: Consider the singular linear second order hyperbolic equation defined on a rectangular domain [120, Example 2]

$$u_{tt} + \frac{2}{x^2}u_t + \frac{1}{x^2}u = (1+x^2)u_{xx} - \frac{e^{-2t}\left(x^4 - 3x^2 + 3\right)\sinh(x)}{x^2},\ 0 < x < 1, 0 < t < T,$$

subject to the initial and boundary conditions (3.31) and (3.32) with exact solution $u(x,t) = e^{-2t}\sinh(x)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig 3.12. The step length used to generate the mesh points for the training and test set are $\frac{1}{10}$ and $\frac{1}{64}$ respectively. The kernel bandwidth parameter used in the simulation is $\sigma = 0.90$.

The obtained results are tabulated in Table 3.8. The proposed method shows a better performance in comparison with the unconditionally stable finite difference scheme of $O(k^2 + h^2)$ described in [120] in terms of accuracy despite the fact that much less number of mesh points are used.

Problem 12: Consider the linear second order elliptic equation defined on a rectangular domain [86, Example 5]

$$\nabla^2 u(x,y) = \exp(-x)(x - 2 + y^3 + 6y)$$

with $x, y \in [0,1]$ and the Dirichlet boundary conditions:

$$u(0,y) = y^3,\ \ u(1,y) = (1+y^3)\exp(-1)$$

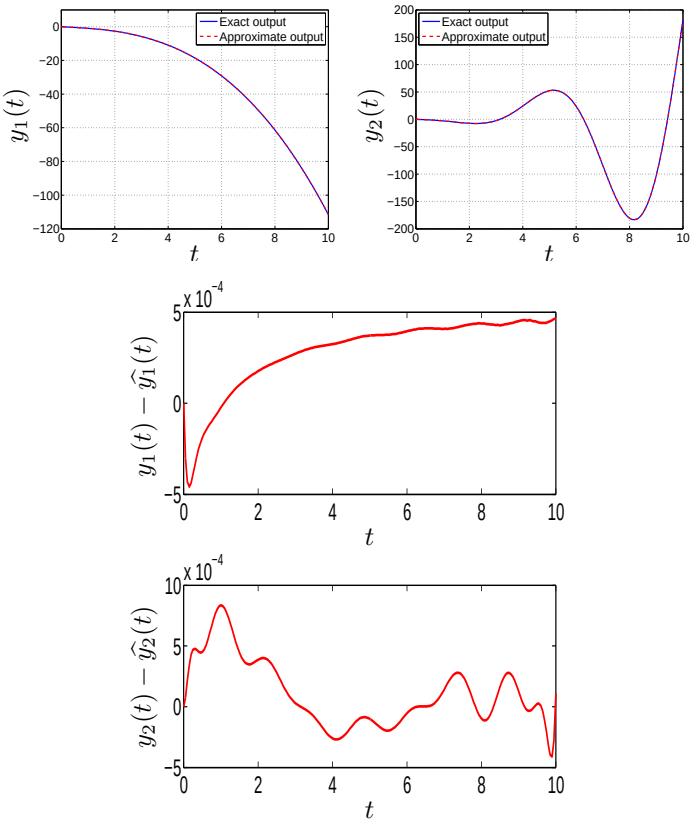


Figure 3.11: Obtained approximate solution and model errors for problem 10, when 50 equidistant mesh points on the interval [0,10] are used for training.

and

$$u(x,0) = x\exp(-x), \quad u(x,1) = x\exp(-x)(x+1)$$

The exact solution is $u(x,y) = e^{-x}(x+y^3)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig 3.13.

In order to make a fair comparison, the same number of grid points as in [86] is used for training and test sets. [86] uses an approach based on Artificial neural networks where a trial solution is provided by the user. The proposed method shows slightly better performance in comparison with the described method in [86] in terms of accuracy (The maximum absolute error for training and test points shown in [86, Fig. 9 and Fig. 10] is approximately 5×10^{-7}).

Table 3.8: Numerical result of the proposed method for solving Problem 11 with time interval $[0, T]$.

		RMSE		$\boldsymbol{L_\infty}$	
Method	**T**	Training	Test	Training	Test
LSSVM	1	3.79×10^{-5}	3.71×10^{-5}	8.38×10^{-5}	9.52×10^{-5}
FDM [120]		– – – – –	0.22×10^{-3}	– – – – –	– – – – –
LSSVM	2	1.78×10^{-5}	1.76×10^{-5}	4.48×10^{-5}	4.89×10^{-5}
FDM [120]		– – – – –	0.62×10^{-4}	– – – – –	– – – – –

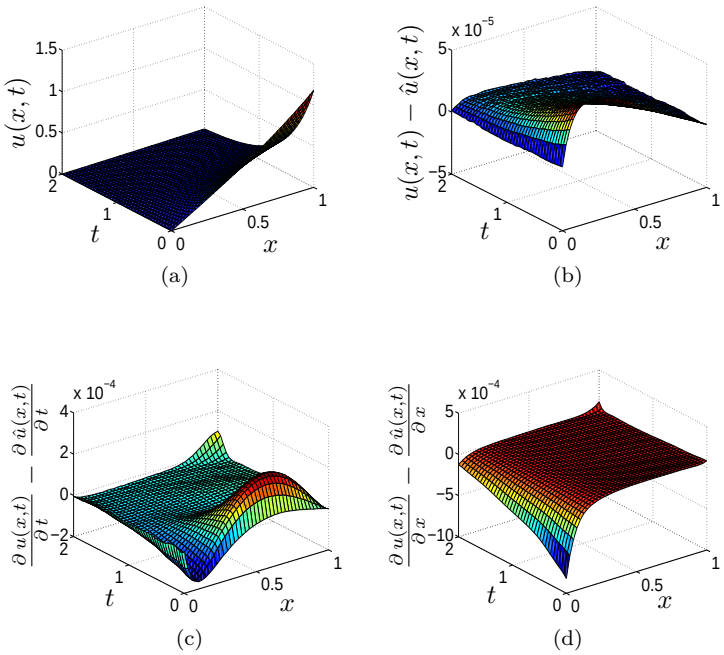


Figure 3.12: Obtained model errors for problem 11, when a grid consists of 171 mesh points inside the domain $[0, 1] \times [0, 2]$ are used for training.

Furthermore as opposed to the neural networks approach here one does not need to provide a trial neural form of the solution and the solution is obtained by solving a linear system of equations. The kernel bandwidth parameter used in the simulation is $\sigma = 0.71$.

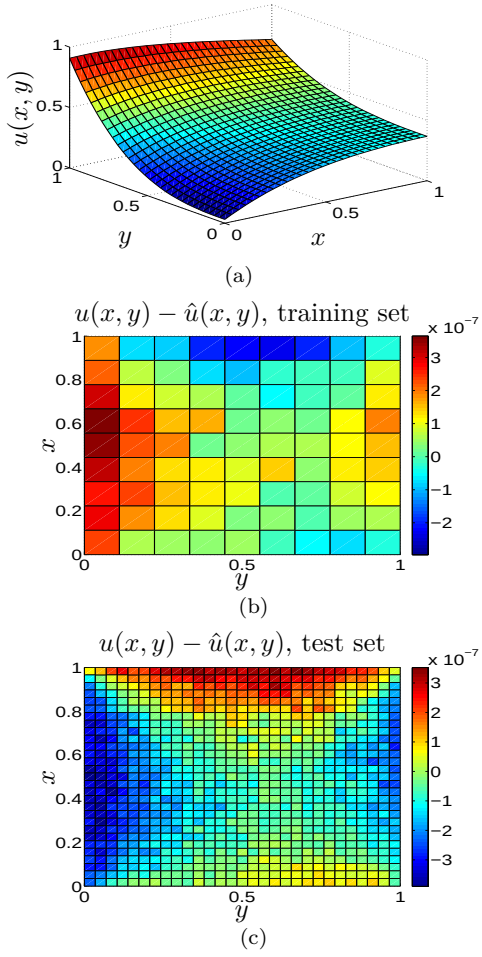


Figure 3.13: The obtained approximate solution and the model errors for problem 12. (a) The obtained approximate solution, (b) The model error on training set when a grid consists of 100 mesh points inside the domain $[0, 1] \times [0, 1]$ are used for training, (c) The model error on test set consists of 900 mesh points inside the domain $[0, 1] \times [0, 1]$ are used for testing.

Problem 13: Consider the linear second order elliptic PDE [149, Section V-B1]

$$\nabla^2 u(x,y) = 4x\cos(x) + (5 - x^2 - y^2)\sin(x) \tag{3.44}$$

defined on a circular domain, i.e.

$$\Sigma := \left\{ (x,y) \,\middle|\, x^2 + y^2 - 1 = 0,\ -1 \le x \le 1, -1 \le y \le 1 \right\}$$

with the Dirichlet condition $u(x,y) = 0$ on $\partial\Sigma$. The exact solution is given by $u(x,y) = (x^2 + y^2 - 1)\sin(x)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig 3.14. The distribution of the collocation points used to undertake the learning process is shown in Fig. 3.3(b). The kernel bandwidth parameter used in the simulation is $\sigma = 3.0$. The number of collocation points (training points) inside and on the boundary of the domain are as follows,

$$|\mathcal{Z}_{\mathcal{D}}| = 45,\ |\mathcal{Z}_{\mathcal{B}}| = 19,$$

which are less than those (52 and 24 collocation points inside and on the boundary of the domain respectively) used in [149]. The obtained results are tabulated in Table 3.9. The proposed method outperforms the described method in [149] in terms of accuracy despite the fact that less training points are used. (Note that in [149] the maximum absolute error shown in [149, Fig. 7] is approximately 2×10^{-3} and the reported mean square error in [149, Table II], obtained by using genetic programming with boosting approach, is 2.05×10^{-4}).

Table 3.9: Numerical result of the proposed method for solving Problem 13 and 14.

		MSE		$\boldsymbol{L_\infty}$	
Problem	**Method**	Training	Test	Training	Test
13	LSSVM	5.18×10^{-11}	5.94×10^{-11}	1.91×10^{-5}	2.71×10^{-5}
	GPA[149]	− − − − −	2.04×10^{-4}	− − − − −	− − − − −
14	LSSVM	7.93×10^{-9}	1.32×10^{-8}	3.95×10^{-4}	5.90×10^{-4}
	GPA[149]	− − − − −	4.46×10^{-4}	− − − − −	− − − − −

Problem 14: Consider the second order elliptic PDE [149, Section V-B2]

$$\nabla^2 u(x,y) = 2\exp(x - y) \tag{3.45}$$

defined on the following domain, i.e.

$$\Sigma := \left\{ (x,y) \,\middle|\, (x,y) = r(\theta)\Big(\cos(\theta), \sin(\theta)\Big),\ 0 \le \theta \le 2\pi, \right\}$$

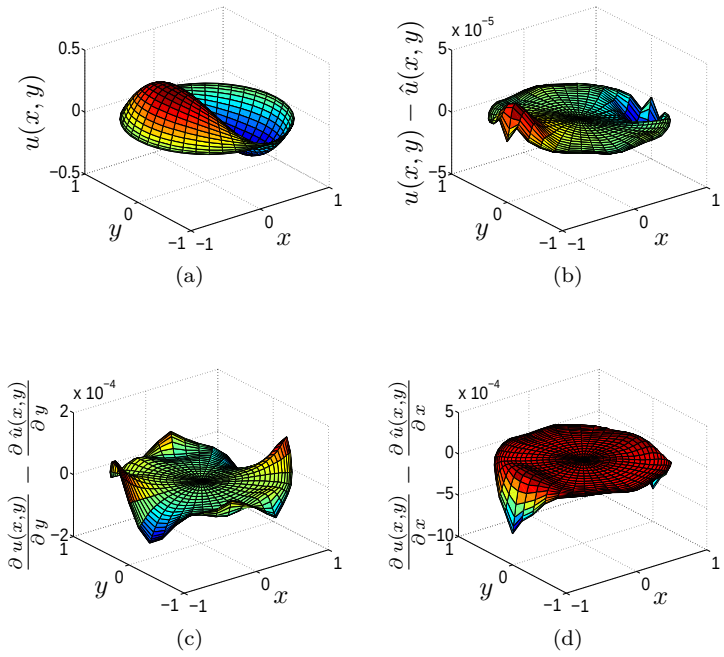


Figure 3.14: Obtained model error for problem 13, when a grid consists of 45 and 19 mesh points inside and on the boundary of the domain respectively are used for training.

with $r(\theta) = \sqrt{\cos(2\theta) + \sqrt{1.1 - \sin^2(2\theta)}}$ and the Dirichlet boundary condition $u(x, y) = e^{x-y} + e^x \cos(y)$ on $\partial\Sigma$. The exact solution is given by $u(x, y) = e^{x-y} + e^x \cos(y)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig 3.15. The distribution of the collocation points used to undertake the learning process is shown in Fig. 3.3(c). The kernel bandwidth parameter used in the simulation is $\sigma = 1.0$. The number of collocation points (training points) inside and on the boundary of the domain are as follows,

$$|\mathcal{X}_{\mathcal{D}}| = 48, \ |\mathcal{X}_{\mathcal{B}}| = 28,$$

which are almost the same as the ones (48 and 32 collocation points inside and on the boundary of the domain respectively) used in [149]. The computed residuals are displayed in Fig 3.15(b)-(d). The mean squared errors and maximum absolute errors for the test set are also recorded in Table 3.9, which

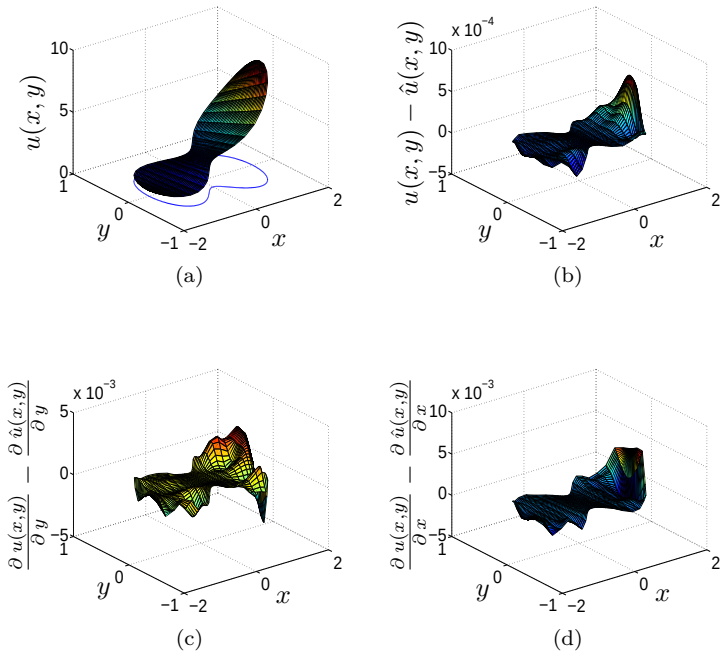


Figure 3.15: Obtained model error for problem 14, when a grid consists of 48 and 28 mesh points inside and on the boundary of the domain respectively are used for training.

shows the improvement of the proposed method over the described method in [149]. (Note that in [149] the maximum absolute error shown in [149, Fig. 11] is approximately 2×10^{-2} and the reported mean square error in [149, Table II], obtained by using genetic programming with boosting approach, is 4.46×10^{-4}).

Problem 15: Consider an example of nonlinear PDE

$$\nabla^2 u(x,y) + u(x,y)^2 = \sin(\pi x)\Big(2 - (\pi y)^2 + y^4 \sin(\pi x)\Big) \tag{3.46}$$

defined on a circular domain, i.e.

$$\Sigma := \Big\{(x,y)\,\Big|\, x^2 + y^2 - 1 = 0,\ -1 \le x \le 1, -1 \le y \le 1\Big\}$$

with the Dirichlet condition on $\partial\Sigma$. The exact solution is given by $u(x, y) = y^2 \sin(\pi x)$. The approximate solution obtained by the proposed method is compared with the exact solution in Fig 3.16(b). The kernel bandwidth parameter used in the simulation is $\sigma = 0.80$. The number of collocation points (training points) inside and on the boundary of the domain are as follows,

$$|\mathcal{Z}_{\mathcal{D}}| = 24, \ |\mathcal{Z}_{\mathcal{B}}| = 19.$$

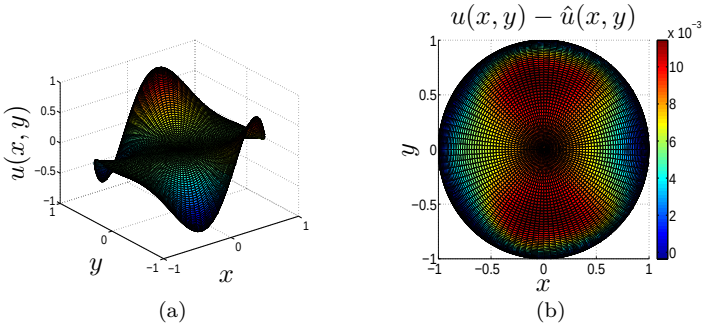


Figure 3.16: Obtained model error for problem 15.

3.8 Conclusions

In this chapter, we presented an LSSVM based formulation for learning the trajectories of a dynamical system whose dynamics is described by either ODEs, DAEs or PDEs. The solution in the primal is in terms of the feature map and the optimal representation is obtained in the dual by solving a set of linear/nonlinear equations. The approach produces a closed form solution and the training points do not need to be uniformly distributed and in fact can be scattered discrete points. As the proposed approach does not require meshing, it has a potential to become an alternative method for learning the solution of high-dimensional PDEs. However the derivation of the dual system still needs user effort. We showed the applicability of the approach for providing the solution of high-index DAEs with variable rank without a need of using any index reduction technique.

Chapter 4

Parameter Estimation of Dynamical Systems

In this chapter, a new approach based on Least Squares Support Vector Machines (LSSVMs) for parameter estimation of time invariant as well as time varying dynamical SISO systems is proposed. Closed-form approximate models for the state and its derivative are first derived from the observed data by means of LSSVMs. The time-derivative information is then substituted into the system of ODEs, converting the parameter estimation problem into an algebraic optimization problem. In the case of time invariant systems one can use least-squares to solve the obtained system of algebraic equations. The estimation of time-varying coefficients in SISO models, is obtained by assuming an LSSVM model for it. Furthermore we extend the approach for approximating time-varying as well as constant parameters in deterministic parameter-affine delay differential equations (DDEs). As opposed to conventional approaches, it avoids iterative simulation of the given dynamical system. The solution obtained by the proposed approach can be further utilized for initialization of the conventional nonconvex optimization methods for parameter estimation of DDEs. The highlights of this chapter can be summarized as follows:

- *Reducing the parameter estimation problem to an algebraic optimization problem.*
- *Avoiding iterative simulation of the dynamical system governed by ODEs, DDEs and delay differential equation of neutral type (NDDEs) in the parameter estimation process.*

- *Providing closed-form approximation for the time varying parameters.*
- *Requiring no prior knowledge about the history function while the fixed delay parameter is being estimated.*
- *Overcoming the non-convexity of the optimization problems for parameter estimation of parameter affine ODEs and DDEs.*

4.1 Related Work

Parameter estimation is widely used in modelling of dynamic processes in physics, engineering and biology. Various methods have been previously investigated in the literature for handling this problem. Mainly they fall into two categories. In the first category the approaches are based on a classical parameter estimator, usually the least square estimator [25]. First the dynamical system is simulated using initial guesses for parameters (if the initial conditions are unavailable they will be appended to the parameters of the model). Then model predictions are compared with measured data and an optimization algorithm updates the parameters. The process of updating the parameters continues until no significant improvement in the objective function is observed. These approaches require numerical integration of differential equations for each update of the parameters. Therefore there is a large amount of computational work involved. Studies show that more than 90% of the computation time is consumed in the ODE solver during the identification process [121].

The second category includes methods, originally proposed by [168], that do not require repeated numerical integration and are referred to as two-step approaches. In [168] first a cubic spline is used to estimate the system dynamics from observational data. The predicted model then can be differentiated with respect to time to obtain the estimate of the derivative of the solution. In the second step these estimates are plugged into a given differential equation and the unknown parameters are found by minimizing the squared difference of both sides of the differential equation.

Identification of unknown parameters in differential equations has been studied and addressed by many authors (see [17, 18, 3, 71, 72, 178]). Most of the available approaches utilize the classical parametric inference such as the least squares estimator or the maximum likelihood estimation [26]. In these approaches first the dynamical system is simulated using initial guesses for the parameters. Then model predictions are compared with measured data and an optimization algorithm updates the parameters.

Therefore one has to solve the following optimization problem:

$$\underset{\theta(t),\tau_1}{\operatorname{argmin}} J(\theta(t),\tau_1) = \sum_{k=1}^{N} (y^m(t_k) - y^p(t_k))^2, \tag{4.1}$$

where $y^m(t)$ and $y^p(t)$ are the measured data and model prediction respectively.

It should be noted that the objective function of the optimization problem for DDE differs from that of ODE. The cost function $J(\theta(t),\tau_1)$ in (4.1) might be non-smooth because the state trajectory might be non-smooth in the parameter and this will make the optimization problem more complicated.

Solving (4.1) requires repeated simulation of the system of DDE under study. Since the analytic solution of DDE is usually not available, therefore one needs to apply a numerical algorithm to simulate the given dynamic system. Although quite efficient numerical routines for solving differential equations are available they usually slow down the parameterization process dramatically and this situation is even more sensible when the underlying dynamics is described by delay differential equations. That is due to the existence of delay terms that force the solver to use an interpolation technique in order to advance the solution. It should also be noted that, as opposed to ordinary differential equations, the numerical solution of DDEs not only depends on the parameter values, but also on the history function, $H_1(t)$ for $t \in [\rho, t_{in}]$, which is usually unknown. Given that the initial function is in an infinite-dimensional set, the problem becomes an infinite-dimensional optimization problem and very difficult to solve [173]. Consequently, it would be of great benefit to eliminate any need of numerical DDE solvers.

The authors in [56] first estimate the derivative $\dot{x}(t)$ from the noisy data using nonparametric smoothing methods and then inferred the constant delay τ, for a special DDE model, in the framework of the generalized additive model. The author in [54] proposed a method where an artificial neural network model is used to estimate the time invariant parameters of a dynamical systems governed by ordinary differential equations. Despite the fact that the classical neural networks have nice properties such as universal approximation, they still suffer from having two persistent drawbacks. The first problem is the existence of many local minima solutions. The second problem is how to choose the number of hidden units. It is the purpose of this chapter to introduce an approach based on least squares support vector machines for estimation of time invariant as well as time varying systems in state-space form.

Throughout this chapter, we assume that the dynamical system is uniquely solvable and that the parameters of the model are identifiable. For stability of the solutions of systems with delays one may refer to [85, 125].

4.2 Dynamical Systems Governed by ODEs

Suppose that we are given a dynamical system in state-space form

$$\frac{dX}{dt} = F(t, X, \theta),\ X(0) = X_0, \tag{4.2}$$

subject to certain boundary or initial conditions which may be imposed on the basis of observation data. t denotes the independent variable (usually time). X is the state vector of the system where $\frac{d}{dt}\hat{X} = [\frac{d}{dt}\hat{x}_1, ..., \frac{d}{dt}\hat{x}_m]^T$, $X = [x_1, ..., x_m]^T$ and $F = [f_1, ..., f_m]^T$. $\theta = [\theta_1, ..., \theta_p]$ are unknown parameters of the system and X_0 are initial values.

In order to estimate the unknown parameters θ, the state variable $X(t)$ is observed at N time instants $\{t_1, ..., t_N\}$, so that we have

$$Y(t_i) = X(t_i) + E_i,\ i = 1, ..., N,$$

where $\{E_i\}_{i=1}^N$ are independent measurement errors with zero mean. The objective is to determine appropriate parameter values so that errors between the outputs of the estimated model and the measured data are minimized.

4.2.1 Constant parameter estimation

First we approximate the trajectory $\hat{X}(t) = [\hat{x}_1, ..., \hat{x}_m]^T$ on the basis of observations at N points $\{t_i, Y(t_i)\}_{i=1}^N$. Note that $Y(t_i)$ are the experimentally observed values of the state variables at time instant t_i, i.e. $Y(t_i) = [y_1(t_i), ..., y_m(t_i)]^T$. Then the estimation of the state derivative is obtained by differentiating the model with respect to time. Here we model the state x_k for $k = 1, ..., m$ as a Least-Squares Support Vector Machine [156]. Therefore the goal is to find a model of the form $\hat{x}_k(t) = w_k^T \varphi(t) + b_k$. For the k-th state variable we formulate the following convex primal LSSVM problem [156],

$$\begin{aligned} \underset{w_k, b_k, e_k}{\text{minimize}} \quad & \frac{1}{2} w_k^T w_k + \frac{\gamma_k}{2} \|e_k\|_2^2 \\ \text{subject to} \quad & y_k(t_i) = w_k^T \varphi(t_i) + b_k + e_k^i, \quad i = 1, ..., N, \end{aligned} \tag{4.3}$$

where $\gamma_k \in \mathbb{R}^+, b_k \in \mathbb{R}$, $w_k \in \mathbb{R}^h$. $\varphi(\cdot) : \mathbb{R} \to \mathbb{R}^h$ is the feature map and h is the dimension of the feature space. The dual solution is then given by

$$\left[\begin{array}{c|c} \Omega + \gamma^{-1} I_N & 1_N \\ \hline 1_N^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha^k \\ \hline b_k \end{array}\right] = \left[\begin{array}{c} y^k \\ \hline 0 \end{array}\right] \tag{4.4}$$

where $\Omega_{ij} = K(t_i, t_j) = \varphi(t_i)^T \varphi(t_j)$ is the (i,j)-th entry of the positive definite kernel matrix. $1_N = [1; ...; 1] \in \mathbb{R}^N$, $\alpha^k = [\alpha_1^k; ...; \alpha_N^k]$, $y^k = [y_k(t_1); ...; y_k(t_N)]$ and I_N is the identity matrix. The model in dual form becomes:

$$\hat{x}_k(t) = w_k^T \varphi(t) + b_k = \sum_{i=1}^{N} \alpha_i^k K(t_i, t) + b_k \tag{4.5}$$

where K is the kernel function. Differentiating (4.5) with respect to t, one can obtain an analytical approximate expression for the derivative of the model

$$\frac{d}{dt}\hat{x}_k(t) = w_k^T \dot{\varphi}(t) = \sum_{i=1}^{N} \alpha_i^k \varphi(t_i)^T \dot{\varphi}(t). \tag{4.6}$$

Making use of Mercer's Theorem [167], derivatives of the feature map can be written in terms of derivatives of the kernel function [92]. Therefore $\varphi(t)^T \dot{\varphi}(s)$ is given by the derivative of $K(t,s)$ with respect to s. If we denote $K_s(t,s) = \frac{\partial K(t,s)}{\partial s}$, then equation (4.6) can be written as

$$\frac{d}{dt}\hat{x}_k(t) = w_k^T \dot{\varphi}(t) = \sum_{i=1}^{N} \alpha_i^k K_s(t_i, t). \tag{4.7}$$

Eqs. (4.5) and (4.7) are closed-form approximations for the k-th state in equation (4.2) and its derivative respectively. By applying the above procedure for all the state variables one can obtain the LSSVM expression for $\hat{X} = [\hat{x}_1, ..., \hat{x}_m]^T$ and $\frac{d}{dt}\hat{X} = [\frac{d}{dt}\hat{x}_1, ..., \frac{d}{dt}\hat{x}_m]^T$. Therefore the values of the solution and time-derivative curves at some set of sample points $\{t_k\}_{k=1}^M$, which are not necessarily the same as the original points where the states are observed, can be obtained by evaluating the LSSVM expressions for $\hat{X}$ and $\frac{d}{dt}\hat{X}$. These numerical values then are substituted into the system description (4.2), so that the unknown parameters appear in an algebraic expression, resulting in linear (if the system is linear in the parameters) or nonlinear (otherwise) least-squares estimation. Therefore the estimation of time invariant parameters is obtained by solving the following optimization problem [111]:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{2}\sum_i \|\Xi_i\|_2^2 \\ \text{subject to} \quad & \Xi_i = \frac{d}{dt}\hat{X}(t_i) - F(t_i, \hat{X}(t_i), \theta), \quad i = 1, ..., M. \end{aligned} \tag{4.8}$$

When the ODE model is linear in the parameters this problem is a convex optimization problem. As it has been remarked in [168], what we really are

interested in to minimize is the error between the observed and model predicted values of the state variables i.e. the integrated residual errors

$$R_I(\theta_{est}) = \sum_{k=1}^{M} \left\| X(t_k) - \tilde{X}(t_k) \right\|_2^2 \tag{4.9}$$

where $\tilde{X}(t_k)$ is obtained by simulating the system with the estimated parameter θ_{est}.

4.2.2 Time varying parameter estimation

Consider a first order dynamical system of the form:

$$\frac{dx}{dt} + \theta(t) f(x(t)) = g(t),\ x(0) = x_0 \tag{4.10}$$

subject to certain initial conditions which may be imposed on the basis of observation data. The technique can be extended to higher order systems. f is an arbitrary known function and $\theta(t)$ is the time varying parameter of the system and is considered to be unknown. The state $x(t)$ has been measured at certain time instants $\{t_i\}_{i=1}^N$, which can be non-equidistant, i.e.

$$y(t_i) = x(t_i) + \xi_i,\ i = 1, ..., N$$

where ξ_i's are i.i.d. random errors with zero mean and constant variance. $g(t)$ is the input signal whose values are known at data points $\{t_i\}_{i=1}^N$ i.e. $g_i = g(t_i)$ for $i = 1, ..., N$. The problem is to estimate the function $\theta(t)$ so that the solution of (4.10) with the estimated parameter $\theta(t)$ is as close as possible to the given data.

First we approximate functions $\hat{x}(t)$ and $\hat{g}(t)$ on the basis of observations at N points $\{t_i, y_i\}_{i=1}^N$, $\{t_i, g_i\}_{i=1}^N$ by means of least squares support vector regression (4.3). The model in dual form becomes,

$$\hat{x}(t) = w^T \varphi(t) + b = \sum_{i=1}^{N} \alpha_i K(t_i, t) + b \tag{4.11}$$

where K is the kernel function. The same procedure can be applied to obtain the LSSVM approximation of the excitation $\hat{g}(t)$. Note that the analytic LSSVM expression for the state trajectory allows us to obtain a closed-form approximation for its derivative by differentiating (4.11) with respect to t,

$$\frac{d}{dt}\hat{x}(t) = w^T \dot{\varphi}(t) = \sum_{i=1}^{N} \alpha_i \varphi(t_i)^T \dot{\varphi}(t) = \sum_{i=1}^{N} \alpha_i K_s(t_i, t). \tag{4.12}$$

Here $K_s(t,s)$ is defined as previously. Eqs. (4.11) and (4.12) are approximations for the solution of the differential equation (4.10) and its derivative respectively. Therefore the derivative of the solution at some set of sample points $\{t_k\}_{k=1}^M$ can be obtained from (4.12). These time-derivative information together with values of the state variable at points $\{t_k\}_{k=1}^M$ are then substituted into the model description (4.10). But since the parameter present in (4.10) is time-varying, it can not be estimated by Eq. (4.8). Therefore let us assume an explicit LSSVM model

$$\hat{\theta}(t) = v^T\psi(t) + b_\theta$$

as an approximation for the parameter $\theta(t)$. Having available the state and its derivative at $\{t_k\}_{k=1}^M$ points, we can estimate the time-varying coefficient $\theta(t)$ by solving the following optimization problem [111]:

$$\begin{aligned} \underset{v,b_\theta,e}{\text{minimize}} \quad & \frac{1}{2}v^Tv + \frac{\gamma}{2}\sum_{i=1}^{M} e_i^2 \\ \text{subject to} \quad & \frac{d}{dt}\hat{x}(t_i) + \Big[v^T\psi(t_i) + b_\theta\Big] f(\hat{x}(t_i)) = \\ & \hat{g}(t_i) + e_i, \text{ for } i = 1, ..., M. \end{aligned} \tag{4.13}$$

Lemma 4.2.1. *Given a positive definite kernel function $\tilde{K} : \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ with $\tilde{K}(t,s) = \psi(t)^T\psi(s)$ and a regularization constant $\gamma\in\mathbb{R}^+$, the solution to (4.13) is given by the following dual problem*

$$\left[\begin{array}{c|c} D\Omega D + \gamma^{-1}I_M & f(\hat{x}) \\ \hline f(\hat{x})^T & 0 \end{array}\right]\left[\begin{array}{c} \alpha \\ \hline b_\theta \end{array}\right] = \left[\begin{array}{c} \hat{g} - \frac{d\hat{x}}{dt} \\ \hline 0 \end{array}\right] \tag{4.14}$$

where $\Omega(i,j) = \tilde{K}(t_i,t_j) = \psi(t_i)^T\psi(t_j)$ is the (i,j)-th entry of the positive definite kernel matrix. Also $\alpha = [\alpha_1; ...; \alpha_M]$, $f(\hat{x}) = [f(\hat{x}(t_1)); ...; f(\hat{x}(t_M))]$, $\hat{g} = [\hat{g}(t_1); ...; \hat{g}(t_M)]$, $\frac{d\hat{x}}{dt} = [\frac{d}{dt}\hat{x}(t_1); ...; \frac{d}{dt}\hat{x}(t_M)]$ and I_M is the identity matrix. D is a diagonal matrix with the elements of $f(\hat{x})$ on the main diagonal.

Proof. The Lagrangian of the constrained optimization problem (4.13) becomes

$$\mathcal{L}(v, b_\theta, e_i, \alpha_i) = \frac{1}{2}\,v^Tv + \frac{\gamma}{2}\sum_{i=1}^{M} e_i^2 -$$

$$\sum_{i=1}^{M}\alpha_i\left[\frac{d}{dt}\hat{x}_i + \Big(v^T\psi(t_i) + b_\theta\Big) f(\hat{x}_i) - \hat{g}_i - e_i\right]$$

where $\{\alpha_i\}_{i=1}^M$ are Lagrange multipliers. $\hat{g}_i = \hat{g}(t_i)$, $f(\hat{x}_i) = f(\hat{x}(t_i))$ and $\frac{d}{dt}\hat{x}_i = \frac{d}{dt}\hat{x}(t_i)$ for $i = 1, ..., M$. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\frac{\partial \mathcal{L}}{\partial v} = 0 \rightarrow v = \sum_{i=1}^{M} \alpha_i f(\hat{x}_i)\psi(t_i),$$

$$\frac{\partial \mathcal{L}}{\partial b_\theta} = 0 \rightarrow \sum_{i=1}^{M} \alpha_i f(\hat{x}_i) = 0,$$

$$\frac{\partial \mathcal{L}}{\partial e_i} = 0 \rightarrow e_i = -\frac{\alpha_i}{\gamma}, \quad i = 1, ..., M,$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow \left(v^T\psi(t_i) + b_\theta\right) f(\hat{x}_i) - e_i = \hat{g}_i - \frac{d}{dt}\hat{x}_i,$$

$$\text{for } \; i = 1, ..., M.$$

After elimination of the primal variables v and $\{e_i\}_{i=1}^M$ and making use of Mercer's Theorem, the solution is given in the dual by

$$\begin{cases} \hat{g}_i - \frac{d}{dt}\hat{x}_i = \sum_{j=1}^{M} \alpha_j f(\hat{x}_j)\Omega_{ji} f(\hat{x}_i) + \frac{\alpha_i}{\gamma} + \\ \qquad\qquad b_\theta f(\hat{x}_i), \; i = 1, ..., M \\ 0 = \sum_{i=1}^{M} \alpha_i f(\hat{x}_i) \end{cases}$$

and writing these equations in matrix form gives the linear system in (4.14). □

The model in the dual form becomes

$$\hat{\theta}(t) = v^T\psi(t) + b_\theta = \sum_{i=1}^{M} \alpha_i f(\hat{x}_i)\tilde{K}(t_i, t) + b_\theta \tag{4.15}$$

where $\tilde{K}$ is the kernel function.

The procedure of the proposed approach is outlined in Algorithm 3.

4.3 LSSVM Based Initialization Approach

Parameter estimation is typically formulated as the following non-convex optimization problem where the multiple-shooting approach (see Fig. 4.1) is

Algorithm 3: Approximating the model's time varying parameter

Input: Observational data and the underlying differential equations
Output: Estimation of the unknown model's parameters

1 Estimate the trajectories $\hat{X}$ from the observational data by using LSSVM model, Eq. (4.5).
2 Differentiate the predicted model with respect to time to get an approximate model for the derivative of the state, Eq. (4.7).
3 Evaluate the state and its derivative model at time instants $\{t_i\}_{i=1}^{M}$.
4 **if** *parameters are time invariant* **then**
5 | solve optimization problem (4.8)
else
6 | solve Eq. (4.14) to get the estimate of the time varying parameter of the dynamical system.

7 **return** *Model parameters*

employed.

$$\underset{X(t_0),\ldots,X(t_N),\theta}{\text{minimize}} \quad \frac{1}{2}\sum_{i=0}^{N} \|Y(t_i) - X(t_i)\|_2^2$$

$$\text{subject to} \qquad X(t_{k+1}) = X(t_k) + \int_{t_k}^{t_{k+1}} F(\tau, X(\tau), \theta)\, d\tau, \quad k = 0, ..., N-1. \tag{4.16}$$

Due to the nonconvexity coming from the nonlinear model, a Newton type method can only find locally optimal solutions. Depending on the initialization, one can obtain a different local solution.

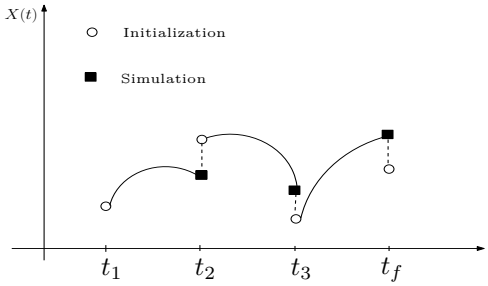


Figure 4.1: The gap between ■ , ○ is minimized when the junction conditions are satisfied.

In case of parameter-affine models attempts have been made to provide a good initial guess through a convex optimization approach. A Least Squares Prediction Error Method (PEM) proposed in [98] is formulated as a convex problem to provide such an initial guess for (4.16), with parameter-affine function F, as follows:

$$\begin{aligned} \underset{X(t_0),\ldots,X(t_N),\theta}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=0}^{N} \|Y(t_i) - X(t_i)\|_2^2 \\ \text{subject to} \quad & X(t_{k+1}) = Y(t_k) + T_s F(t_k, Y(t_k), \theta), \quad k = 0, \ldots, N-1, \end{aligned} \tag{4.17}$$

where T_s is the sampling time. However the solution obtained by the PEM approach can be biased if the process and measurement noise are not modeled appropriately. Therefore the method does not perform well in the presence of noisy data and one needs to filter the residual errors. On the other hand, the authors in [28] proposed a so-called Least Squares Convex Approach (CA). In contrast to the PEM approach, there is no need for filtering the residual error in CA formulation [28]:

$$\begin{aligned} \underset{X(t_0),\ldots,X(t_N),\theta}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=0}^{N} \|Y(t_i) - X(t_i)\|_2^2 \\ \text{subject to} \quad & X(t_{k+1}) = X(t_k) + T_s F(t_k, Y(t_k), \theta), \quad k = 0, \ldots, N-1. \end{aligned} \tag{4.18}$$

However in this approach in order to keep the optimization problem (4.20) convex, one still has to rely on a simple Euler discretization of the system.

The aim of this section is to first employ the method described in section 4.2 to obtain an initial guess for the parameters and then to solve the original non-convex problem. The latter is done using a multiple shooting discretization and constrained Gauss-Newton to solve the nonlinear programming problem (NLP). Furthermore, a denoising scheme using LSSVM is proposed to first filter the measured data then proceed with the filtered signals for parameter estimation problem. As opposed to the previous approaches, one does not need to use any integration method to simulate the dynamical system. Therefore the drawbacks of using the Euler method, concerning its stability region, are removed. We refer further to this approach as LSSVM (see [111]).

4.4 Pre-processing using LSSVM

Data pre-processing plays a very important role in many applications. We will make use of the LSSVM regression ability to reduce the effect of noise and as a result having a smoother signal to proceed with. Given observational data $Y(t) = [y_1(t), \ldots, y_m(t)]^T$, the LSSVM based model (4.5) obtained from (4.3) can be considered as a denoised version of the observational data $Y(t)$. The idea now would be to replace the measurements $Y(t)$ in PEM and CA formulations (4.19) and (4.20) by $\widehat{X}(t)$ where $\widehat{X}(t) = [\hat{x}_1(t), \ldots, \hat{x}_m(t)]^T$. Therefore their optimization problems become [111]:

$$\begin{aligned} \underset{X(t_0),\ldots,X(t_N),\theta}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=0}^{N} \|Y(t_i) - X(t_i)\|_2^2 \\ \text{subject to} \quad & X(t_{k+1}) = \widehat{X}(t_k) + T_s F(t_k, \widehat{X}(t_k), \theta), \quad k = 0, ..., N-1, \end{aligned} \tag{4.19}$$

and

$$\begin{aligned} \underset{X(t_0),\ldots,X(t_N),\theta}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=0}^{N} \|Y(t_i) - X(t_i)\|_2^2 \\ \text{subject to} \quad & X(t_{k+1}) = X(t_k) + T_s F(t_k, \widehat{X}(t_k), \theta), \quad k = 0, ..., N-1, \end{aligned} \tag{4.20}$$

These schemes will be referred to as PEM+LSSVM (4.19) and CA+LSSVM (4.20) respectively.

4.5 Dynamical Systems Governed by DDEs

Delay differential equations (DDEs) have been successfully used in the mathematical formulation of real life phenomena in a wide variety of applications especially in science and engineering such as population dynamics, infectious diseases, control problems, secure communication, traffic control and economics [22, 20, 81]. In contrast with ordinary differential equations (ODEs) where the unknown function and its derivatives are evaluated at the same time instant, in a DDE the evolution of the system at a certain time instant, depends

on the state of the system at an earlier time. A typical first order single-delay scalar DDE model may be expressed as:

$$\begin{aligned} \dot{x}(t) &= f_1(t, x(t), x(t-\tau_1), \theta(t)), \quad t \geq t_{in}, \\ x(t) &= \mathcal{H}_1(t), \quad \rho \leq t \leq t_{in} \end{aligned} \tag{4.21}$$

where $\mathcal{H}_1(t)$ is the initial function (history function), τ_1 is the delay or lag which is non-negative and can in general be constant, time dependent or state dependent i.e. $\tau_1 = \tau_1(t, x(t))$ and $\rho = \min_{t \geq t_{in}} \{t - \tau_1\}$. The term $x(t-\tau_1)$ is called the delay term. In more general models, the derivative $\dot{x}(t)$ may depend on $x(t)$ and $\dot{x}(t)$ itself at some past value $t - \tau_1$. In this case equation (4.21) can be rewritten in a more general form as follows

$$\begin{aligned} \dot{x}(t) &= f_2(t, x(t), x(t-\tau_1), \dot{x}(t-\tau_2), \theta(t)), \quad t \geq t_{in}, \\ x(t) &= \mathcal{H}_2(t), \quad \rho \leq t \leq t_{in} \end{aligned} \tag{4.22}$$

where $\rho = \min_{1 \leq i \leq 2} \{\min_{t \geq t_{in}} (t - \tau_i)\}$. Equation (4.22) is called delay differential equation of neutral type (NDDE). Models (4.21) and (4.22) usually involve some unknown parameters that require to be estimated from the observational data. We consider sets $\{\theta(t), \mathcal{H}_1(t), \tau_1\}$ and $\{\theta(t), \mathcal{H}_2(t), \tau_1, \tau_2\}$ as parameters of the models (4.21) and (4.22) respectively.

4.5.1 Problem statement

Here we consider two cases, the fixed delays are unknown or there is an unknown time varying parameter in the system. Next the precise problem statements are described.

4.5.1.1 Reconstruction of fixed delays

Consider the dynamics of a process during a given time interval modeled by a system of nonlinear DDEs with associated history functions $\mathcal{H}(t)$ of the form:

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), x(t-\tau_1), x(t-\tau_2), \ldots, x(t-\tau_p)), \quad t \geq t_{in}, \\ x(t) &= \mathcal{H}(t), \quad \rho \leq t \leq t_{in} \end{aligned} \tag{4.23}$$

where $\rho = \min_{1 \leq i \leq p} \{\min_{t \geq t_{in}} (t - \tau_i)\}$, $x(t) \in \mathbb{R}^n$ and the delays $\{\tau_i\}_{i=1}^p$ are constant and unknown. In order to estimate the model parameters, all the states of the

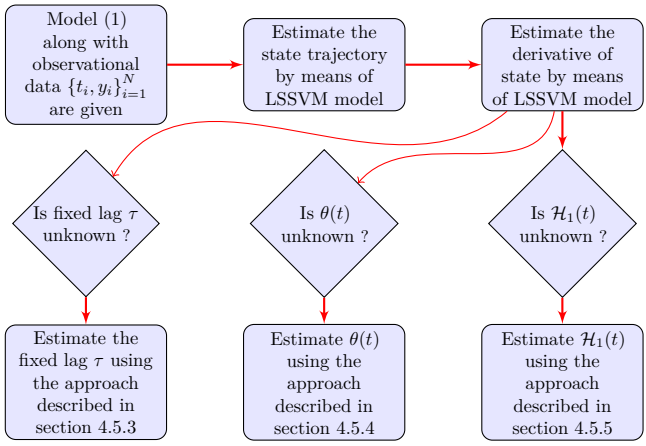
system are measured i.e. $y(t_i) = x(t_i) + e(t_i)$ where $\{e(t_i)\}_{i=1}^{N}$ are independent measurement errors with zero mean. Here a particular structure of (4.23) is considered. It is assumed that nonlinear model (4.23) exhibits the parameter-affine form i.e. it is affine in the $x(t - \tau_i)$ for $i = 1, \ldots, p$.

4.5.1.2 Reconstruction of time varying parameters

Consider the nonlinear state-dependent delay differential equation given in (4.21) with associated history function $\mathcal{H}_1(t)$. In order to estimate the unknown parameters, a set of measurements $y(t_i)$ are collected. In general the set of measurements $y(t_i)$ do not necessarily correspond to the model states $x(t_i)$. However here it is assumed that the system states are measured with measurement error $e(t_i)$, therefore the sate space model has the following form:

$$\begin{aligned} \dot{x}(t) &= f_1(t, x(t), x(t-\tau_1), \theta(t)), \;\; t \geq t_{in}, \\ y(t_i) &= x(t_i) + e_i, \; i = 1, \ldots, N \end{aligned} \tag{4.24}$$

where $y(t)$ is the output of the system which has been observed at N time instants and $\{e_i\}_{i=1}^{N}$ are independent measurement errors with zero mean. The unknown $\{\mathcal{H}_1(t), \theta(t)\}$ are time dependent. In order to keep the model affine in the unknown time varying parameters we do not assume that both of them are unknown at the same time. Therefore as in [71, 72], we consider the case that one of them is unknown at the time of applying the estimation procedure. Hence the following cases can be studied: (i) $\mathcal{H}_1(t)$ is known and $\theta(t)$ is unknown, (ii) $\theta(t)$ is known and the history function $\mathcal{H}_1(t)$ is unknown, The same assumption is made for parameter estimation of the neutral delay differential equation (4.22). The general stages of the procedure when the dynamic system follows model (4.21) is described by the following flow-chart:



4.5.2 General Methodology

The proposed scheme will make use of the LSSVM ability to provide a closed-form approximation for the state trajectory and its derivative from measured data. We approximate the trajectory $\hat{x}(t)$ on the basis of observations at N points $\{t_i, y(t_i)\}_{i=1}^N$ using (4.5). Then (4.7) is utilized for approximating the state derivative. These closed-form expressions will be used later in the process of parameter estimation.

4.5.3 Fixed delay τ is unknown

For the sake of simplicity the methodology is described for a scalar DDE with single delay, but the approach is applicable for identifying multi-delays in a system of DDEs provided that they are identifiable. Consider the following single delay parameter-affine DDE:

$$\dot{x}(t) = f(t, x(t))x(t-\tau), \quad t \geq t_{in}, \tag{4.25}$$

where $f(\cdot) : \mathbb{R}^2 \longrightarrow \mathbb{R}$ is an arbitrary nonlinear function and τ is the constant parameter of the system which is unknown. In order to estimate the unknown τ value, the state of the system is measured i.e. $y(t_i) = x(t_i) + e(t_i)$ where $\{e(t_i)\}_{i=1}^N$ are independent measurement errors with zero mean. Let us assume an explicit LSSVM model

$$\hat{x}_\tau(t) = v^T \psi(t) + d,$$

as an approximation for the term $x(t-\tau)$ where $\psi(\cdot) : \mathbb{R} \to \mathbb{R}^h$ is the feature map. Substituting the closed-form expressions for the state and its derivative, $\frac{d}{dt}\hat{x}(t)$ and $\hat{x}(t)$ obtained from (4.5) and (4.7) respectively, into the model description (4.25), the sought parameters v and d are identified as those minimizing the following optimization problem [114]:

$$\begin{aligned} &\underset{v,d,e}{\text{minimize}} \quad \frac{1}{2}v^T v + \frac{\gamma}{2}\sum_{i=1}^{M} e_i^2 \\ &\text{subject to} \quad \frac{d}{dt}\hat{x}(t_i) = \Big(v^T\psi(t_i) + d\Big) f(t_i, \hat{x}(t_i)) + e_i, \text{ for } i = 1, ..., M. \end{aligned} \tag{4.26}$$

Remark 4.5.1. *Since closed-form expressions for the state and its derivative are available we are not limited to choose $M = N$, i.e. we can evaluate the constraint of the above optimization problem at the time instant t_i which is not necessarily the same as time instants that the system is measured.*

Lemma 4.5.1. *Given a positive definite kernel function $\tilde{K} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with $\tilde{K}(t,s) = \psi(t)^T\psi(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (4.26) is given by the following dual problem*

$$\left[\begin{array}{c|c} D\tilde{\Omega}D + \gamma^{-1}I & F \\ \hline F^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline d \end{array}\right] = \left[\begin{array}{c} \frac{d\hat{x}}{dt} \\ \hline 0 \end{array}\right] \tag{4.27}$$

where $\tilde{\Omega}(i,j) = \tilde{K}(t_i, t_j) = \psi(t_i)^T\psi(t_j)$ is the (i,j)-th entry of the positive definite kernel matrix and I is the identity matrix. Also $\alpha = [\alpha_1, \ldots, \alpha_M]^T$, $F = [f(t_1, \hat{x}(t_1)), \ldots, f(t_M, \hat{x}(t_M))]^T$, $\frac{d\hat{x}}{dt} = [\frac{d}{dt}\hat{x}(t_1), \ldots, \frac{d}{dt}\hat{x}(t_M)]^T$. D is a diagonal matrix with the elements of F on the main diagonal.

Proof. The Lagrangian of the constrained optimization problem (4.26) becomes

$$\mathcal{L}(v, d, e_i, \alpha_i) = \frac{1}{2}\, v^T v + \frac{\gamma}{2}\sum_{i=1}^{M} e_i^2 - \sum_{i=1}^{M} \alpha_i \bigg[\Big(v^T\psi(t_i) + d\Big) f(t_i, \hat{x}(t_i)) + e_i - \frac{d}{dt}\hat{x}(t_i)\bigg],$$

where $\{\alpha_i\}_{i=1}^M$ are Lagrange multipliers. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\frac{\partial \mathcal{L}}{\partial v} = 0 \rightarrow v = \sum_{i=1}^{M} \alpha_i f(t_i, \hat{x}(t_i))\psi(t_i),$$

$$\frac{\partial \mathcal{L}}{\partial d} = 0 \rightarrow \sum_{i=1}^{M} \alpha_i f(t_i, \hat{x}(t_i)) = 0,$$

$$\frac{\partial \mathcal{L}}{\partial e_i} = 0 \rightarrow e_i = \frac{\alpha_i}{\gamma}, \quad i = 1, \ldots, M,$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow \left(v^T \psi(t_i) + d \right) f(t_i, \hat{x}(t_i)) + e_i = \frac{d}{dt}\hat{x}(t_i), \text{for } \; i = 1, \ldots, M.$$

After elimination of the primal variables v and $\{e_i\}_{i=1}^M$ and making use of Mercer's Theorem, the solution is given in the dual by

$$\begin{cases} \frac{d}{dt}\hat{x}(t_i) = \sum_{j=1}^{M} \alpha_j f(t_j, \hat{x}(t_j))\Omega_{ji} f(t_i, \hat{x}(t_i)) + \frac{\alpha_i}{\gamma} + df(t_i, \hat{x}(t_i)), \; i = 1, \ldots, M \\ 0 = \sum_{i=1}^{M} \alpha_i f(t_i, \hat{x}(t_i)) \end{cases}$$

Writing these equations in matrix form gives the linear system in (4.27). □

The model in the dual form becomes

$$\hat{x}_\tau(t) = v^T \psi(t) + d = \sum_{i=1}^{M} \alpha_i f(t_i, \hat{x}(t_i))\tilde{K}(t_i, t) + d, \tag{4.28}$$

where $\tilde{K}$ is the kernel function.

Remark 4.5.2. *If one is not interested in having a closed-form approximation to the term $x(t-\tau)$, an alternative way to obtain an approximation for $x(t-\tau)$ at the time instant t_i is by using (4.25) directly, i.e. $x(t_i - \tau) = \frac{d}{dt}\hat{x}(t_i)^{-1} f(t_i, \hat{x}(t_i))$. A similar strategy can be applied in the case that the dynamics of the process is described by a system of delay differential equations. After substituting the closed-form expressions for the states and their derivatives into the model, then one has to solve a system of linear equations (provided that the underlying system is affine in the unknown parameter) to obtain the approximation of the delay terms $x(t-\tau_j)$ for $j = 1, \ldots, p$ at time instants $t = t_i$, for $i = 1, \ldots, N$.*

After obtaining the estimation $\hat{x}_\tau(t)$, the task is to estimate the fixed delay τ. To this end, let us first define a shifting operator $\Delta_m(\cdot)$ which will be used in the process of estimation of the delay τ. Operator $\Delta_m(\cdot)$ shifts the given time series, which in our problem setting can for example be $\hat{x}(t)$ or $\hat{\dot{x}}(t)$, m steps forward in time in a certain manner, while keeping the length of the time series unchanged. This is done by adding a constant vector of size m (whose values will be clarified later) from the left to the time series and removing the m last elements of the time series simultaneously. Therefore, given the time series $\hat{x}(t) = [\hat{x}(t_1), \hat{x}(t_2), \ldots, \hat{x}(t_N)]^T$, operator $\Delta_m(\cdot)$ is defined as follows:

$$z(t) = \Delta_m(\hat{x}(t)) = \begin{cases} [\underbrace{z(t_1), \ldots, z(t_m)}_{\text{Constant vector}}, \hat{x}(t_1), \ldots, \hat{x}(t_{N-m})]^T, & 1 \leq m \leq N-1 \\ \hat{x}(t), & \text{for } m = 0 \end{cases} \tag{4.29}$$

with $z(t_1) = z(t_2) = \ldots, z(t_m) = c$ where c is a constant. Noting that in an ideal case (noise free) one can expect a delay differential equation to have the following property

$$\hat{x}_\tau(t)\Big|_{t=\tau} = \hat{x}(t)\Big|_{t=t_{in}}, \quad \text{for } \tau \geq 0, \tag{4.30}$$

it is natural to utilize the first element of $\hat{x}(t)$, i.e., $\hat{x}(t_1)$ as a constant c used in operator $\Delta_m(\cdot)$. In order to estimate the delay τ, we use the sample correlation coefficient function defined as:

$$r_{z\hat{x}_\tau} = \frac{\sum_{i=1}^{N}(z(t_i) - \mu_1)(\hat{x}_\tau(t_i) - \mu_2)}{\sqrt{\sum_{i=1}^{N}(z(t_i) - \mu_1)^2}\sqrt{\sum_{i=1}^{N}(\hat{x}_\tau(t_i) - \mu_2)^2}}, \tag{4.31}$$

where μ_1 and μ_2 denote the sample mean of time series $z(t)$ and $\hat{x}_\tau(t)$ respectively. Given $\hat{x}(t)$ and $\hat{x}_\tau(t)$ the process of estimating the unknown delay τ is described in Algorithm 4.

Algorithm 4: Approximating the constant delay of a given DDE

Input: Time series $\hat{x}(t)$ and $\hat{x}_\tau(t)$ of size N; sampling time T_s (in seconds).
Output: Time delay τ

for $m \leftarrow 0$ ***to*** $N-1$ **do**
  $z(t) \leftarrow \Delta_m(\hat{x}(t))$
  $R(m) \leftarrow \texttt{Corrcoef}(z(t), \hat{x}_\tau(t))$
$\tau \leftarrow T_s \times \underset{m}{\text{argmax}}\, R(m)$
return τ

In Algorithm 4, `Corrcoef` is a Matlab built-in function that computes the correlation coefficient of two signals and $R(m)$ corresponds to $r_{z\hat{x}_\tau}$. One may notice that in this approach we are not using the history function for estimating the time delay τ. But if the history function is known a priori, one may use it for constructing the constant vector used in operator $\Delta_m(\cdot)$ by taking the value of history function at time t_{in}.

4.5.4 Parameter $\theta(t)$ is unknown

Consider model (4.21) and case (i) where the time varying parameter $\theta(t)$ is unknown and delay τ_1 is known. Therefore with a slight abuse of notation, let us assume an explicit LSSVM model

$$\hat{\theta}(t) = v^T\psi(t) + d,$$

as an approximation for the parameter $\theta(t)$. The adjustable parameters v and d are to be found by solving the following optimization problem [114]:

$$\begin{aligned}
\underset{v,d,e,\epsilon,\theta_i}{\text{minimize}} \quad & \frac{1}{2}v^Tv + \frac{\gamma}{2}(\sum_{i=1}^{M} e_i^2 + \sum_{i=1}^{M}\epsilon_i^2) \\
\text{subject to} \quad & \frac{d}{dt}\hat{x}(t_i) = f_1(t_i, \hat{x}(t_i), \hat{x}(t_i-\tau_1), \theta_i) + e_i, \text{ for } i = 1,\ldots,M, \\
& \theta_i = v^T\psi(t_i) + d + \epsilon_i, \text{ for } i = 1,\ldots,M.
\end{aligned} \tag{4.32}$$

Here the obtained closed-form expressions for the state and its derivative, $\frac{d}{dt}\hat{x}(t)$ and $\hat{x}(t)$ obtained from (4.5) and (4.7), are substituted into the model description (4.21). If f_1 is nonlinear in $\theta(t)$ then the above optimization problem is non-convex. The solution of (4.32) in the dual can be obtained by solving a system of nonlinear equations. However, here we present our results for the case that the nonlinear model (4.21) is affine in the parameter $\theta(t)$. More precisely we consider the following parameter-affine form of (4.21)

$$\begin{aligned}
&\dot{x}(t) = \theta(t)f_1(t, x(t), x(t-\tau_1)), \quad t \geq t_{in}, \\
&x(t) = \mathcal{H}_1(t), \quad t \leq t_{in}.
\end{aligned}$$

This will result in the following convex optimization problem:

$$\begin{aligned} &\underset{v,d,e}{\text{minimize}} \quad \frac{1}{2}v^T v + \frac{\gamma}{2}\sum_{i=1}^{M} e_i^2 \\ &\text{subject to} \quad \frac{d}{dt}\hat{x}(t_i) = \Big(v^T\psi(t_i) + d\Big) f_1(t_i, \hat{x}(t_i), \hat{x}(t_i - \tau_1)) + e_i, \text{ for } i = 1, \ldots, M. \end{aligned} \tag{4.33}$$

Lemma 4.5.2. *Given a positive definite kernel function $\tilde{K} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ with $\tilde{K}(t,s) = \psi(t)^T\psi(s)$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to (4.33) is given by the following dual problem*

$$\left[\begin{array}{c|c} D\tilde{\Omega}D + \gamma^{-1}I & F_1 \\ \hline F_1^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline d \end{array}\right] = \left[\begin{array}{c} \frac{d\hat{x}}{dt} \\ \hline 0 \end{array}\right] \tag{4.34}$$

where $\tilde{\Omega}(i,j) = \tilde{K}(t_i, t_j) = \psi(t_i)^T\psi(t_j)$ is the (i,j)-th entry of the positive definite kernel matrix and I is the identity matrix. Also $\alpha = [\alpha_1, \ldots, \alpha_M]^T$, $F_1 = [f_1(t_1, \hat{x}(t_1), \hat{x}(t_1 - \tau_1)), \ldots, f_1(t_M, \hat{x}(t_M), \hat{x}(t_M - \tau_1))]^T$, $\frac{d\hat{x}}{dt} = [\frac{d}{dt}\hat{x}(t_1), \ldots, \frac{d}{dt}\hat{x}(t_M)]^T$. D is a diagonal matrix with the elements of F_1 on the main diagonal.

The model in the dual form becomes:

$$\hat{\theta}(t) = \sum_{i=1}^{M} \alpha_i f_1(t_i, \hat{x}(t_i), \hat{x}(t_i - \tau_1))\tilde{K}(t_i, t) + d, \tag{4.35}$$

where $\tilde{K}$ is the kernel function.

Proof. The approach is the same as in proof of Lemma 4.5.1. □

It should be noted that in the process of estimating $\theta(t)$, the values of the history function $\mathcal{H}_1(t)$ are not used. Therefore $\mathcal{H}_1(t)$ can also be unknown while $\theta(t)$ is being estimated which is the advantage of the proposed method compared with conventional approaches that require the history function for simulating the underlying model.

Remark 4.5.3. *The same procedure can be applied for estimating the unknown parameter $\theta(t)$ in parameter-affine form of model (4.22).*

4.5.5 History function $\mathcal{H}_1(t)$ is unknown

Consider model (4.21) and case (ii) where the parameter $\mathcal{H}_1(t)$ is unknown and all the other parameters are known. It is assumed that the nonlinear function f_1 is affine in $x(t-\tau_1)$. More precisely we consider the following form of (4.21):

$$\begin{aligned} \dot{x}(t) &= x(t-\tau_1) f_1(t, x(t), \theta(t)), \quad t \geq t_{in}, \\ x(t) &= \mathcal{H}_1(t), \quad t \leq t_{in} \end{aligned} \tag{4.36}$$

where τ_1 can be time and state dependent. Since the history function is time varying let us, with a slight abuse of notation, assume an explicit LSSVM model

$$\hat{\mathcal{H}}_1(t) = v^T \psi(t) + d,$$

as an approximation to the true $\mathcal{H}_1(t)$. Optimal value for v and d can be obtained by solving the following convex optimization problem [114]:

$$\begin{aligned} \underset{v,d,e}{\text{minimize}} \quad & \frac{1}{2} v^T v + \frac{\gamma}{2} \sum_{i=1}^{|\mathcal{T}|} e_i^2 \\ \text{subject to} \quad & \frac{d}{dt}\hat{x}(t_{sel}^i) = \Big(v^T \psi(t_{sel}^i) + d\Big) f_1(t_{sel}^i, \hat{x}(t_{sel}^i), \theta(t_{sel}^i)) + e_i, \\ & \text{for } i = 1, \ldots, |\mathcal{T}|, \end{aligned} \tag{4.37}$$

where $\frac{d}{dt}\hat{x}(t_{sel}^i)$ and $\hat{x}(t_{sel}^i)$ are estimations of the state trajectory and its derivative obtained by using LSSVM models (4.5) and (4.7) respectively. $|\mathcal{T}|$ is the cardinality of the ordered set $\mathcal{T} = \{t_{sel}^1, t_{sel}^2, \ldots, t_{sel}^{|\mathcal{T}|}\}$ whose elements are selected using Algorithm 5.

Algorithm 5: Approximating the model's time varying history function

Input: Vector T consists of time instants $\{t_i\}_{i=1}^N$ and the delay τ_1
Output: set $\mathcal{T}$

1 **for** $i \leftarrow 1$ ***to*** N **do**
2 $\quad tlag(i) \leftarrow t_i - \tau_1(t_i)$
3 **end**
4 Find a vector of indices of elements of *tlag* whose values are less than t_{in} (assuming that $t_{in} = t_1$)
5 $\mathcal{T} \leftarrow$ elements of T corresponding to the indices found in step 2.
6 **return** $\mathcal{T}$

The solution to (4.37) in the dual can be obtained by solving linear system (4.34) with $\alpha = [\alpha_1, \ldots, \alpha_{|\mathcal{T}|}]^T$, $F_1 = [f_1(t_{sel}^1, \hat{x}(t_{sel}^1), \theta(t_{sel}^1)), \ldots, f_1(t_{sel}^{|\mathcal{T}|}, \hat{x}(t_{sel}^{|\mathcal{T}|}), \theta(t_{sel}^{|\mathcal{T}|}))]^T$

and $\frac{d\hat{x}}{dt} = [\frac{d}{dt}\hat{x}(t^1_{sel}), \ldots, \frac{d}{dt}\hat{x}(t^{|\mathcal{T}|}_{sel})]^T$. D is a diagonal matrix with the elements of F_1 on the main diagonal. The model in the dual form becomes:

$$\hat{\mathcal{H}}_1(t) = \sum_{i=1}^{|\mathcal{T}|} \alpha_i f_1(t^i_{sel}, \hat{x}(t^i_{sel}), \theta(t^i_{sel}))\tilde{K}(t_i, t) + d, \tag{4.38}$$

where $\tilde{K}$ is the kernel function. If delay τ_1 in the model (4.36) is constant, one can first utilize Algorithm 1 to estimate the delay τ_1 and then apply Algorithm 2 to obtain a closed-form approximation to the history function $\mathcal{H}_1(t)$.

Remark 4.5.4. *The same procedure can be applied for estimating the unknown history function $\mathcal{H}_2(t)$ in a parameter-affine form of model (4.22).*

4.6 Experiments

4.6.1 Parameter estimation of ODEs

To illustrate the applicability of the proposed method, we list the computed results of the parameter estimation for three systems with time invariant coefficients and two first order systems with time varying parameter. For all the experiments, the RBF kernel is used, i.e. $K(x, y) = \exp(-\frac{(x-y)^2}{\sigma^2})$.

4.6.1.1 Constant parameters

Example 1. Consider the nonlinear Bellman's problem originated from a chemical reaction [23]

$$\frac{dx}{dt} = \theta_1(126.2 - x)(91.9 - x)^2 - \theta_2 x^2, x(1) = 0. \tag{4.39}$$

The observations of the state x with one decimal place accuracy are given in Table 4.1.

Table 4.1: Observations of state x for Bellman's problem (4.39) [168].

t	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
x	0.0	1.4	6.3	10.4	14.2	17.6	21.4	23.0
t	10	12	15	20	25	30	40	
x	27.0	30.4	34.4	38.8	41.6	43.5	45.3	

Cross-validation is used to tune the regularization constant γ and kernel bandwidth σ, on a meaningful grid of possible (γ, σ) combinations. The estimated parameter values obtained by averaging over 50 simulation runs and the corresponding integrated residual R_I are as follows

$$[\hat{\theta}_1, \hat{\theta}_2, R_I] = [0.45 \times 10^{-6}, 0.28 \times 10^{-3}, 1.45]$$

which agree well with the true solution $[\theta_1, \theta_2] = [0.45 \times 10^{-6}, 0.27 \times 10^{-3}]$. The standard deviation of our approach for the parameters θ_1 and θ_2 are 8.92×10^{-8} and 1.30×10^{-5} respectively. It should be noted that in the described approach in [168] the spline knots have been chosen interactively. Whereas in our proposed method one does not need to work with the knots and instead the regularization constant γ is chosen automatically to avoid overfitting. Therefore in contrast with the approach of [168] in our proposed method less human effort is needed.

Example 2. Consider Barne's problem which is based on the Lotka-Voltra differential equations consisting of two ordinary differential equations with three parameters θ_1, θ_2 and θ_3 [168]

$$\frac{dx_1}{dt} = \theta_1 x_1 - \theta_2 x_1 x_2,\ x_1(0) = x_{10}$$

$$\frac{dx_2}{dt} = \theta_2 x_1 x_2 - \theta_3 x_2,\ x_2(0) = x_{20}.$$

The observed data values as given by [168] are reported in Table 4.2.

Table 4.2: Observations of the states x_1 and x_2 for Barne's problem [168].

t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
x_1	1.0	1.1	1.3	1.1	0.9	0.7	0.5	0.6	0.7	0.8	1.0
x_2	0.3	0.35	0.4	0.5	0.5	0.4	0.3	0.25	0.25	0.3	0.35

The estimated parameter values are obtained by taking the average over 50 simulation runs where each run corresponds to different training and validation sets. Table 4.3, shows the values of the parameters reported in [168], [147], Matlab *diffpar* [55] toolbox and the computed results obtained by the proposed method. It can be seen that they all are in good agreement.

The standard deviation of our approach for the parameters θ_1, θ_2 and θ_3 are 6.78×10^{-2}, 1.83×10^{-1} and 1.69×10^{-1} respectively. In our approach the $R_I(\theta_{est}) = 0.11$ which is also less than that (i.e. $R_I^2 = 0.35$) reported in [168].

Table 4.3: Estimated parameters of Barne's problem.

Method	$\widehat{\theta_1}$	$\widehat{\theta_2}$	$\widehat{\theta_3}$	$\widehat{x_{10}}$	$\widehat{x_{20}}$
[168]	0.85	2.13	1.91	1.02	0.25
[55]	0.81	2.29	2.00	0.99	0.21
[147]	0.98	1.95	1.69	0.96	0.29
LSSVM approach	0.84	2.14	1.96	0.99	0.29

Furthermore, we have used the approach presented in section 4.3 to estimate the unknown parameters of the Barne's equations and the obtained results are shown in Fig. 4.2.

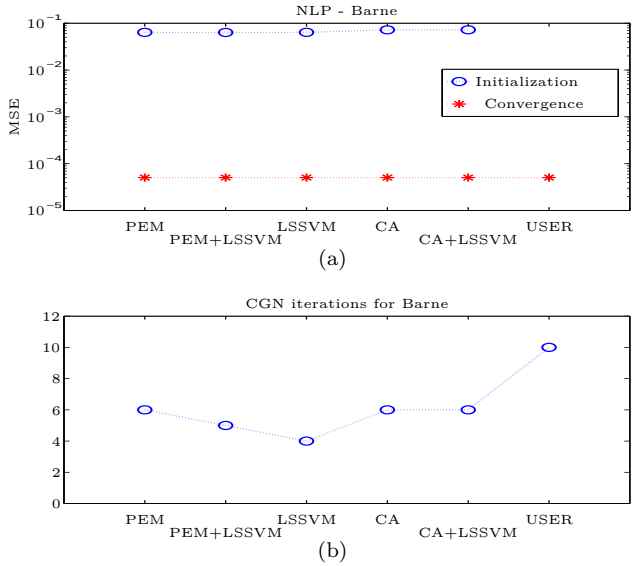


Figure 4.2: The Barne's system is first simulated over domain [0, 200] using the true parameter values. Sampling time $T_s = 1s$ is used i.e. 200 measurements. Gaussian white noise, with zero mean and std 0.1, is added to the true solution. (a) The obtained Mean Squared Error for Barne's model. (b) Number of required Newton iterations needed to satisfy the given tolerance.

In Fig. 4.2, the case where the user is providing some starting point, based upon available prior knowledge, is referred to as USER initialization approach. The implementations and simulations were carried out in MATLAB and for the discretization of (4.16), the ACADO integrators were used as presented in [135].

Fig. 4.2 shows that LSSVM based initialization requires the least number of iterations to converge.

Example 3. Consider the Lorenz equation [99] which form a system of three differential equations that are important in climate and weather predictions. It is well known that the Lorenz equation is an example of a nonlinear and chaotic system

$$\frac{dx_1}{dt} = a(x_2 - x_1)$$

$$\frac{dx_2}{dt} = x_1(b - x_3) - x_2$$

$$\frac{dx_3}{dt} = x_1 x_2 - cx_3$$

where a, b and c are the unknown parameters within the system. The initial condition at $t = 0$ is taken to be $(x_1(0), x_2(0), x_3(0)) = (-9.42, -9.34, 28.3)$. The correct parameters we are trying to reconstruct are $a = 10$, $b = 28$ and $c = 8/3$. The solution of the Lorenz system is prepared by numerically integrating the Lorenz equations using MATLAB built-in solver ode45, on domain [0,3] with the relative error tolerance RelTol= 10^{-6}. Then the model observation data are constructed by adding Gaussian white noise with zero mean to the true solution. The level of noise (standard deviation of the noise) is denoted by η. In this problem η is considered to be $\eta = 0.0, 0.2$ and 0.5. The observation points are prepared within the domain of [0, 3] at every $\Delta t = 0.05$. After obtaining the closed-form approximation for the states x_1, x_2 and x_3 by means of LSSVM, we used 301 equally spaced sample points in the interval [0, 3] to solve optimization problem (4.8). Table 4.4 reports the estimated parameters of the Lorenz system by averaging over 50 simulation runs. The average and standard deviation of our results after 50 simulations are depicted in Fig 4.3.

Table 4.4: The values of parameters estimated of Lorenz model. Parameter η is the standard deviation of the noise.

η	**Estimated parameters**					
	a		b		c	
	a_{est}	$\lvert e_a \rvert$	b_{est}	$\lvert e_b \rvert$	c_{est}	$\lvert e_c \rvert$
0.0	9.99	0.0014	28.00	0.0062	2.67	0.0041
0.2	9.60	0.3919	28.03	0.0352	2.67	0.0042
0.5	9.34	0.6532	27.86	0.112	2.68	0.0147

The true values of model parameters a, b and c are 10.0, 28.0 and 8/3 respectively. Absolute errors are denoted by $|e_.|$.

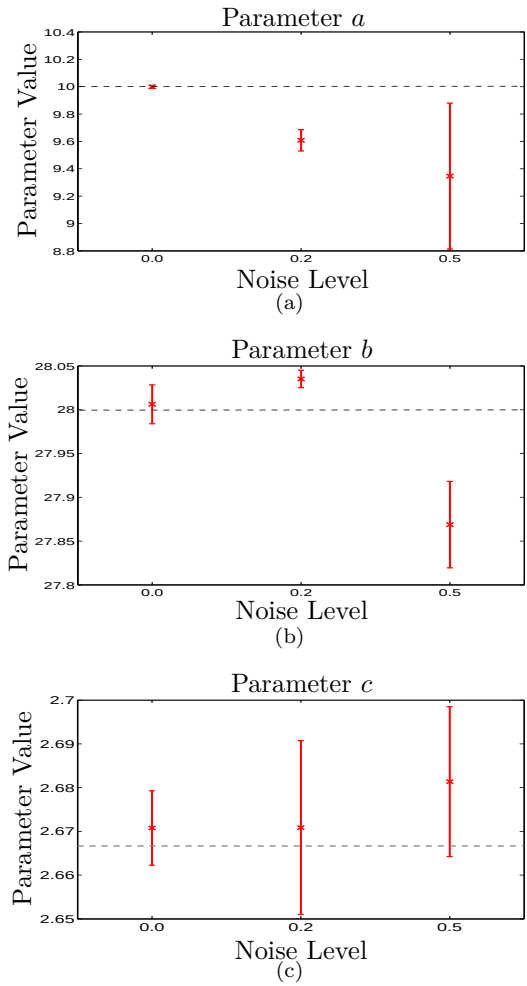


Figure 4.3: Estimation of Lorenz parameters a, b and c from data with observational noise. The true value is indicated by the dashed line.

Furthermore, we have used the approach presented in section 4.3 to estimate the unknown parameters of the Lorenz equations and the obtained results are shown in Fig. 4.4. Fig. 4.4 shows that LSSVM based initialization is comparable to that of conventional convex initialization approaches.

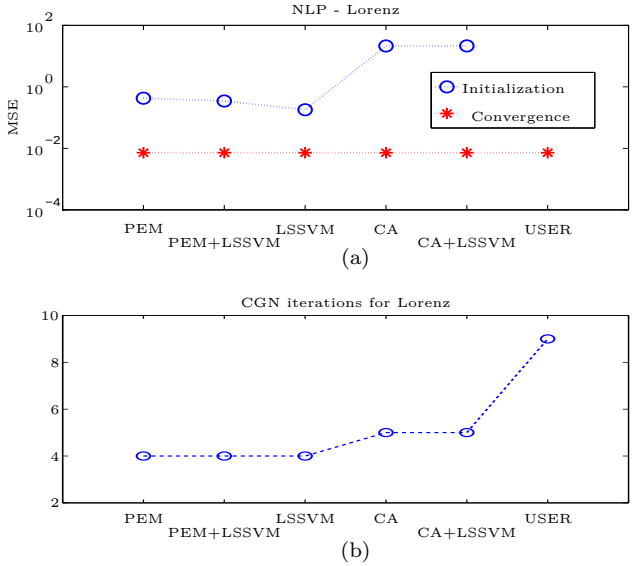


Figure 4.4: The Lorenz system is first simulated over domain [0, 12] using the true parameter values. Sampling time $T_s = 0.04s$ is used i.e. 300 measurements. Gaussian white noise, with zero mean and std 0.05, is added to the true solution. (a) The obtained Mean Squared Error for Lorenz model.(b) Number of required Newton iterations needed to satisfy the given tolerance.

4.6.1.2 Time varying parameters

Example 4. Consider the following linear time varying system,

$$\frac{dx}{dt} - \frac{\sin(t)}{t+1}x(t) = \cos(t), \quad x(0) = 1. \tag{4.40}$$

The aim is to estimate the time varying coefficient $\frac{\sin(t)}{t+1}$ from measured data. For collecting the data, the solution of this equation has been obtained using Matlab built-in solver ode45, with the relative error tolerance RelTol= 10^{-6}, over the domain of [0, 20]. Thereafter the process is observed at N discrete time instants. Then we have artificially introduced random noise (Gaussian white

noise with noise level η) to the true solution in order to create observational data. After obtaining the LSSVM closed form approximation for the state x, we used $M = 200$ equally spaced sample points in the interval [0, 20] to solve optimization problem (4.13). The mean square error (MSE) for the test set (500 sample points in interval [0,20]) are tabulated in Table 4.5. Fig 4.5, shows the influence of noise level on the parameter estimation. 10-fold cross-validation is used for the model selection by choosing one of several models that has the smallest estimated generalization error.

Table 4.5: The influence of noise level and number of observed data on the parameter estimation. Parameter η is the standard deviation of the noise and N is the number of observed data.

N	η	**MSE**
100	0.0	2.02×10^{-8}
	0.1	3.9×10^{-4}
200	0.0	1.09×10^{-8}
	0.1	3.4×10^{-4}

Example 5. Consider the following nonlinear and time varying dynamical system,

$$\frac{dx}{dt} - \frac{\cos(t)}{\sin(t)+2}\cos(x(t)^2) = \cos(t), \quad x(0) = 1. \tag{4.41}$$

In order to estimate the time-varying coefficient $\frac{\cos(t)}{\sin(t)+2}$, we generate the solution to (4.41) with the true parameter. Then random noise (Gaussian white noise with noise level η) is added to the true solution in order to create observational data. The mean square error (MSE) for the test set (500 sample points in interval [0,20]) is tabulated in Table 4.6. Fig 4.6, shows the influence of noise level on the parameter estimation. The kernel bandwidth σ and regularization constant γ are tuned by 10-fold cross validation, while the preference is given to less complex models.

Table 4.6: The influence of noise level and number of observed data on the parameter estimates. Parameter η is the standard deviation of the noise and N is the number of observed data.

N	η	**MSE**
100	0.0	8.34×10^{-5}
	0.05	3.5×10^{-3}
200	0.0	3.06×10^{-6}
	0.05	2.0×10^{-3}

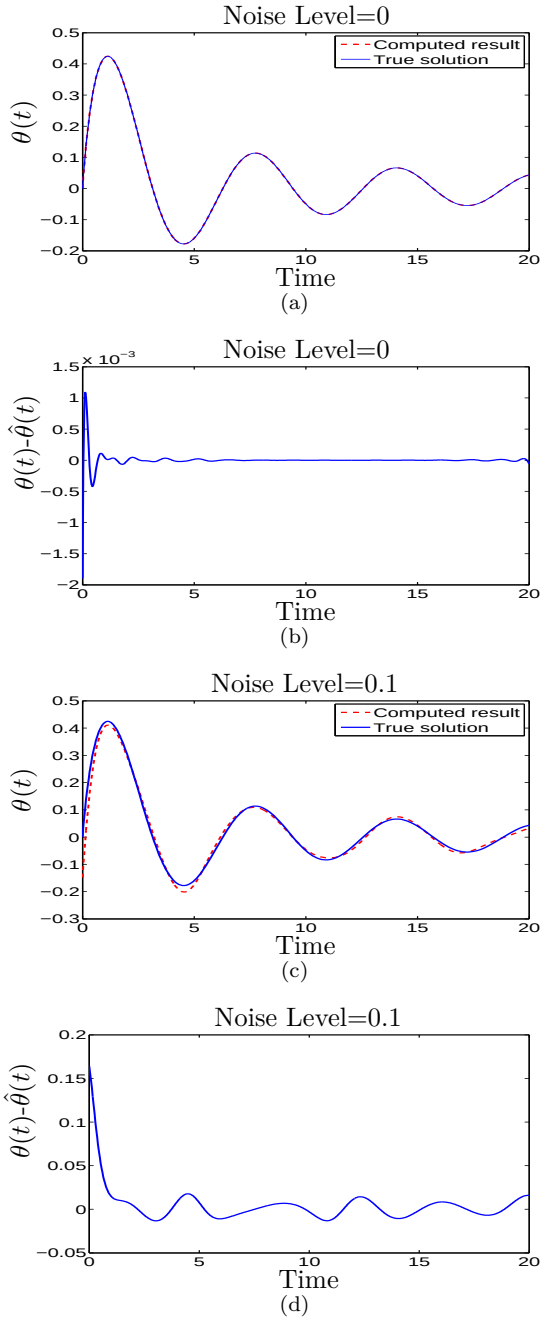


Figure 4.5: Estimation of time varying parameter of dynamical system formulated in Example 4 using $M = 200$ sample points. 500 sample points in the interval $[0, 20]$ are used for the test set.

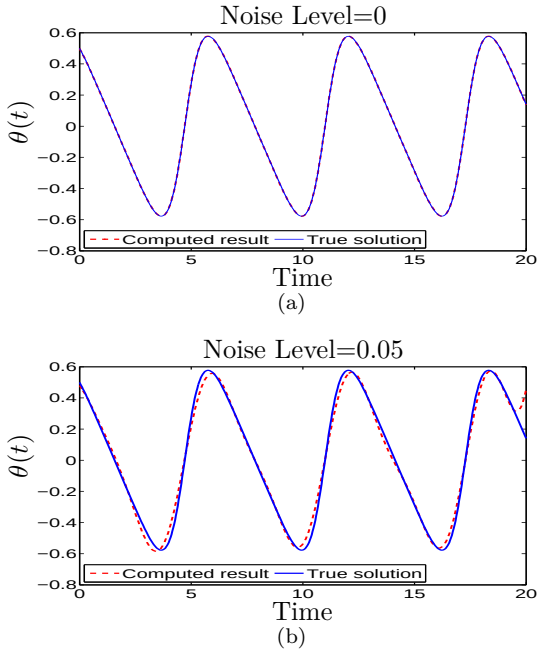


Figure 4.6: Estimation of time varying parameter of dynamical system formulated in Example 5 using $M = 200$ sample points. 500 sample points in the interval $[0, 20]$ are used for the test set.

4.6.2 Parameter estimation of DDEs

In this section, six experiments are performed to demonstrate the capability of the proposed method for time varying/invariant parameters of parameter-affine non-neutral DDEs and neutral DDEs. The last three test problems are taken from [71] and [72], but in contrast with the approach given in these references, we allow to have measurement errors. The performance of the LSSVM model depends on the choice of the tuning parameters. For all experiments, the Gaussian RBF kernel i.e. $K(x, y) = \exp(-\frac{\|x-y\|_2^2}{\sigma^2})$ is used. Therefore, a model is determined by the regularization parameter γ and the kernel bandwidth σ. The 10-fold cross validation criterion is used to tune these parameters. The SNR stands for signal to noise ratio which is calculated using $20 \log_{10}(\frac{A_{signal}}{A_{noise}})$ where A_{signal} and A_{noise} are the root mean square of the signal and noise respectively. The estimated parameter values are obtained by averaging over 10 simulation runs. As error bounds we used about twice the standard deviation of the error.

4.6.2.1 Constant parameters

Problem 6. Consider a Kermack-McKendrick model of an infectious disease with periodic outbreak [143, Example 1]

$$\begin{aligned}\dot{x}_1(t) &= -x_1(t)x_2(t-\tau_1) + x_2(t-\tau_2)\\ \dot{x}_2(t) &= x_1(t)x_2(t-\tau_1) - x_2(t)\\ \dot{x}_3(t) &= x_2(t) - x_2(t-\tau_2)\end{aligned} \tag{4.42}$$

on $[0, 20]$ with history $x_1(t) = 5, x_2(t) = 0.1$ and $x_3(t) = 1$ for $t \leq 0$. The true value of the delays are $\tau_1 = 1$ and $\tau_2 = 10$. For collecting the data, the solution of the this system is prepared by numerically integrating the differential equation (4.42) using MATLAB built-in solver *dde23*, on domain $[0, 20]$ with the relative error tolerance RelTol= 10^{-6}. Then the model observation data are constructed by adding Gaussian white noise with zero mean to the true solution. The observation data points are prepared within the domain of $[0, 20]$ with sampling time $T_s = 100$ ms (i.e. 201 data points). The obtained results are shown in Fig. 4.7. As Fig. 4.7(e) and (g) suggest the peaks of the correlation coefficients occurred nearly at indices 10 and 100. Multiplying these indices with sampling time T_s (in seconds), yields an estimate of the unknown delays τ_1 and τ_2, respectively. Fig. 4.8(a) and (b) show the influence of noise level on the parameter estimation. It should be noted that as the value of signal to noise ratio increases, the standard deviation of the estimation error decreases.

Problem 7. Consider a triangle wave defined by the following scalar NDDE:

$$\begin{aligned}\dot{x}(t) &= -\dot{x}(t-\tau)\\ x(t) &= t,\ -\tau \leq t \leq 0.\end{aligned} \tag{4.43}$$

In order to prepare the observational data, the solution to (4.43) is generated, with the true delay $\tau = 1$, by using MATLAB built-in solver *ddesd*, on domain [0,2] with the relative error tolerance RelTol= 10^{-6}. Then the model observation data are constructed by adding Gaussian white noise with zero mean to the true solution. The observation points are prepared within the domain of $[0, 2]$ with sampling time $T_s = 10$ ms (i.e. 201 data points). Fig. 4.9 represents the results obtained by applying the proposed method for estimating the unknown delay τ. The result of parameter estimation for different values of signal to noise ratio is depicted in Fig. 4.8(c). From Fig. 4.8(c), one may notice that as the value of signal to noise ratio increases, the standard deviation of the estimation error decreases.

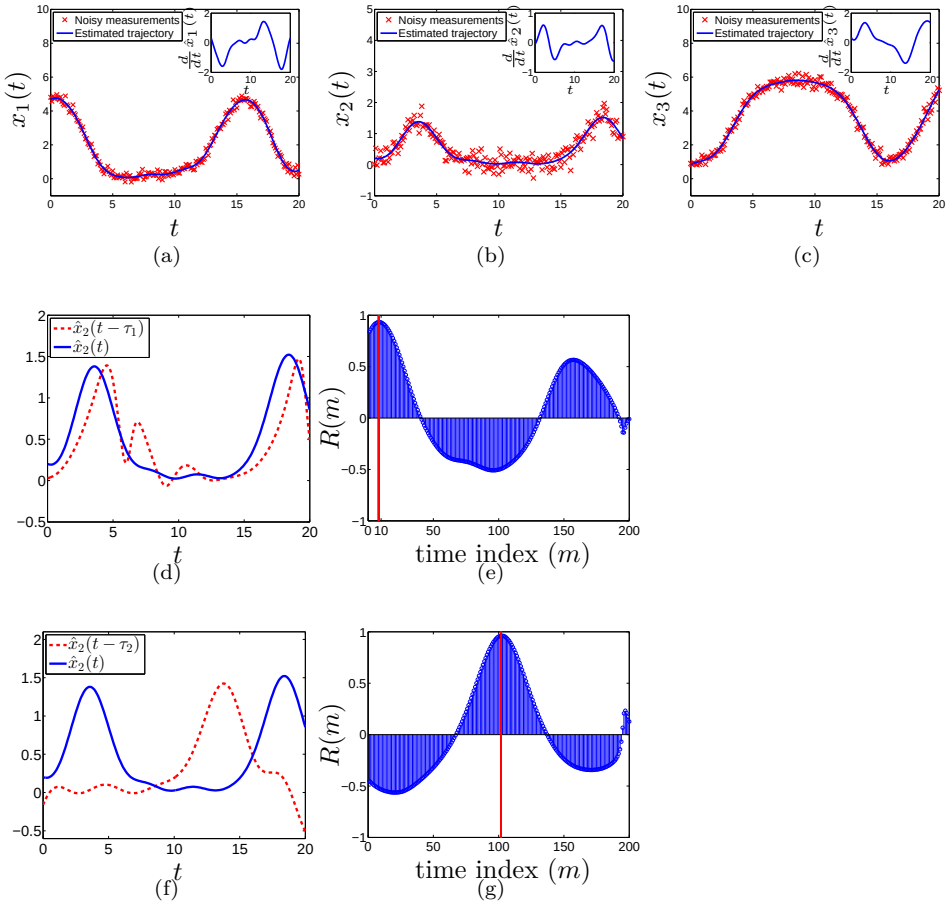


Figure 4.7: Estimation of constant delays τ_1 and τ_2 in Problem 6 from observational data. (a) Estimation of the first state $x_1(t)$ and its derivative from the observational data. (b) Estimation of the second state $x_2(t)$ and its derivative from observational data. (c) Estimation of the third state $x_3(t)$ and its derivative from observational data. (d) Estimation of $x_2(t-\tau_1)$ and $x_2(t)$. (e) Correlation-coefficient values as a function of time index m for two time series $x_2(t)$ and $x_2(t-\tau_1)$ as computed in Algorithm 4. (f) Estimation of $x_2(t-\tau_2)$ and $x_2(t)$. (g) Correlation-coefficient values as a function of time index m for two time series $x_2(t)$ and $x_2(t-\tau_2)$, as computed in Algorithm 4.

Problem 8. Consider an artificial example:

$$\begin{aligned} \dot{x}(t) &= \sin(x(t)+t)x(t-\tau), \quad t \in [0,2] \\ x(0) &= 1, \end{aligned} \tag{4.44}$$

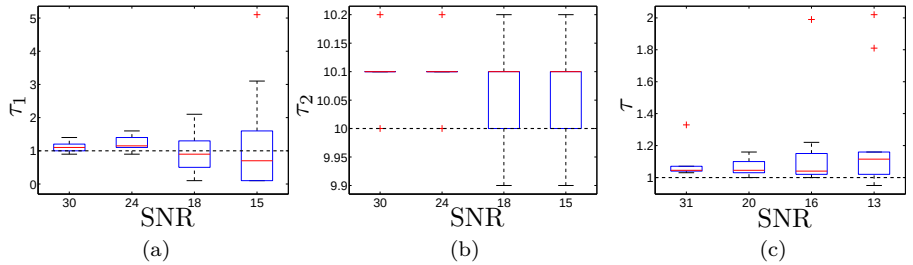


Figure 4.8: Estimation of constant delays τ_1 and τ_2 in Problem 6 and delay τ in Problem 7 from observational data for different values of signal to noise ratio. The exact value of the lags are denoted by the dashed lines. (a) Estimation of delay τ_1 for problem 6. (b) Estimation of the delay τ_2 for problem 6. (c) Estimation of the delay τ for problem 7.

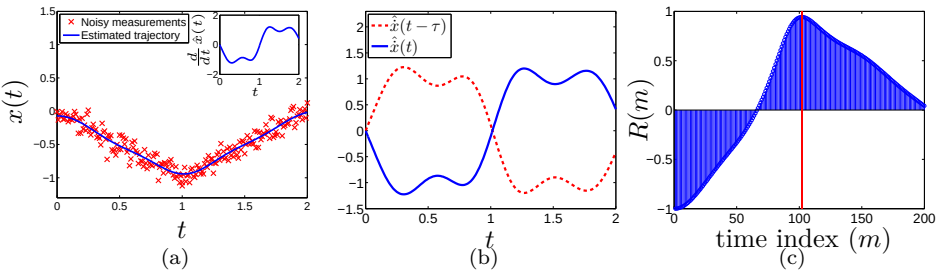


Figure 4.9: Estimation of constant lag τ in Problem 7 from observational data. (a) Estimation of the state $x(t)$ and its derivative from observational data. (b) Estimation of $\hat{\hat{x}}(t-\tau)$ and $\hat{\hat{x}}(t)$. (c) Correlation-coefficient values as a function of time index m for two time series $\hat{\hat{x}}(t)$ and $\hat{\hat{x}}(t-\tau)$, as computed in Algorithm 4.

where the true delay $\tau = 0$. The solution to (4.44) is generated, with the true delay $\tau = 0$, by using MATLAB built-in solver *ode45*, on domain [0,2] with the relative error tolerance RelTol= 10^{-6}. Then the model observation data are constructed by adding Gaussian white noise with zero mean to the true solution. The observation points are prepared within the domain of $[0, 2]$ with sampling time $T_s = 10$ ms (i.e. 201 data points). The obtained results for estimating the unknown delay τ are shown in Fig. 4.10. As Fig. 4.10(c) suggests the peak of the correlation coefficient occurred at index $m = 0$. Based on Algorithm 1, multiplying this index with sampling time T_s (in seconds), yields an estimate

of the unknown delays τ. Thus the estimated lag τ is zero.

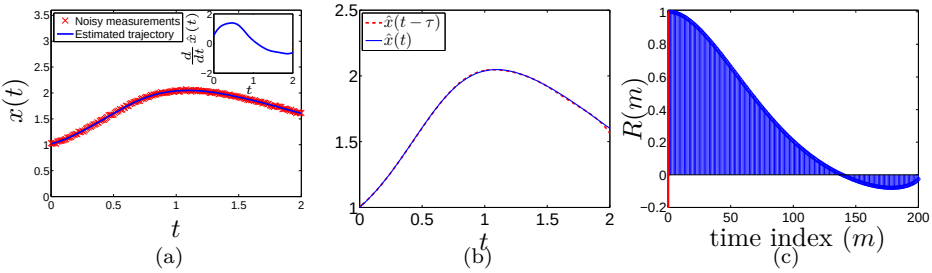


Figure 4.10: Estimation of constant lag τ in Problem 8 from observational data. (a) Estimation of the state $x(t)$ and its derivative from observational data. (b) Estimation of $\hat{x}(t-\tau)$ and $\hat{x}(t)$. (c) Correlation-coefficient values as a function of time index m for two time series $\hat{x}(t)$ and $\hat{x}(t-\tau)$, as computed in Algorithm 4.

4.6.2.2 Time varying parameters

Problem 9. Consider the linear delay equation [71, Problem 2]

$$\begin{aligned} \dot{x}(t) &= \theta(t)x(t-\xi(t)),\ t \in [0,2] \\ x(t) &= \mathcal{H}_1(t),\ t \in [-2,0] \end{aligned} \tag{4.45}$$

where

$$\xi(t) = \begin{cases} 2-t^2, & t \in [0,1] \\ 1, & t \in [1,2] \end{cases}, \quad \theta(t) = \begin{cases} \frac{-t}{t+1}, & t \in [0,1] \\ -\frac{1}{2}, & t \in [1,2] \end{cases}$$

and $\mathcal{H}_1(t) = t^2$. It is assumed that the initial function $\mathcal{H}_1(t)$ and $\xi(t)$ are known and we aim at estimating the unknown parameter $\theta(t)$ from measured data. For collecting the data, the solution of this system is prepared by numerically integrating the differential equation (4.45) using MATLAB built-in solver *ddesd*, on domain [0,2] with the relative error tolerance RelTol= 10^{-6}. Then the model observation data are constructed by adding Gaussian white noise with zero mean to the true solution. The observation points are prepared within the domain of $[0,2]$ with sampling time $T_s = 10$ ms (i.e. 201 data points). Applying the presented scheme in section 4.3, an estimation $\hat{\theta}(t)$ is obtained and the results are depicted in Fig. 4.11(a) and (b). The root mean square errors (RMSE) for different values of signal to noise ratio are also tabulated in Table 4.7. From Table 4.7, it is apparent that as the value of signal to noise ratio (SNR) increases, the estimation error decreases.

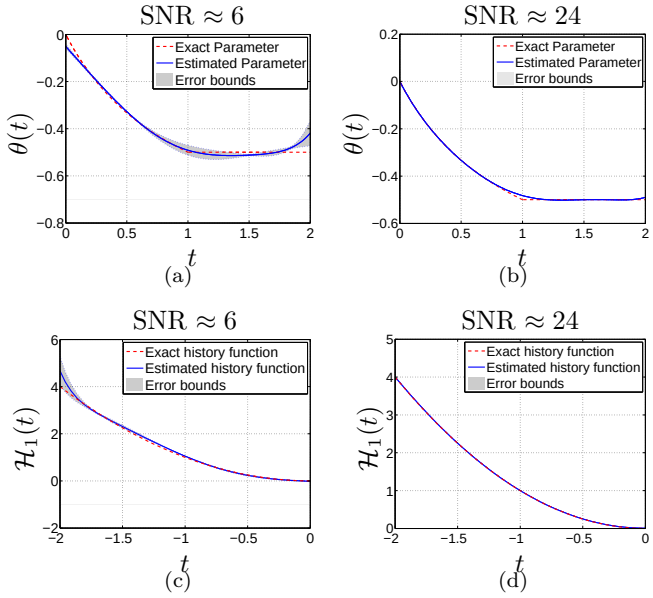


Figure 4.11: (a) and (b) Estimation of time varying parameter $\theta(t)$ in Problem 9 from observational data for different values of signal to noise ratio. (c) and (d) Estimation of History function $\mathcal{H}_1(t)$ in Problem 10 from observational data for different values of signal to noise ratio.

Problem 10. Consider the linear delay equation (4.45). In this problem we assume that $\theta(t)$ and $\xi(t)$ are known and we aim at estimating the initial function from measured data [71, Problem 1]

$$\begin{aligned} \dot{x}(t) &= \theta(t)x(t-\xi(t)),\ t \in [0,2] \\ x(t) &= \mathcal{H}_1(t),\ t \in [-2,0]. \end{aligned} \tag{4.46}$$

As in Problem 9, the observational data are prepared within the domain of $[0,2]$ with sampling time $T_s = 10$ ms (i.e. 201 data points). Fig. 4.11(c) and (d), shows the obtained approximation $\hat{\mathcal{H}}_1(t)$ for the history function when the scheme described in subsection 4.5.5 is utilized. The root mean square errors (RMSE) for different values of signal to noise ratio are recorded in Table 4.7. From Table 4.7, it is apparent that as the value of signal to noise ratio (SNR) increases, the estimated parameter converges to the true parameter.

Table 4.7: The influence of signal to noise ratio on the parameter estimates for problems 9, 10, 11 when 201 data points is used.

	RMS Error		
SNR	**Problem 5.4**	**Problem 5.5**	**Problem 5.6**
6	$1.72e-2$	$2.87e-1$	$1.13e-1$
11	$1.32e-2$	$2.12e-2$	$1.17e-2$
18	$7.01e-3$	$4.02e-3$	$3.14e-3$
24	$2.10e-3$	$2.03e-3$	$1.01e-3$

SNR stands for signal to noise ratio.

Problem 11. Consider the following state dependent delay neutral delay differential equations [72, Problem 1]

$$\begin{aligned} \dot{x}(t) &= \theta(t) + \dot{x}(t - \frac{t^2}{t^2+4}|x(t)| - 1),\ t \in [0,1] \\ x(t) &= \frac{1}{4}t^2 + 1,\ t \leq 0. \end{aligned} \tag{4.47}$$

It is assumed that the time varying parameter $\theta(t)$ is unknown and has to be estimated from measured data. The true parameter is $\theta(t) = \frac{1}{8}t^2 + \frac{1}{2}$. It is easy to check that the true solution of for the given $\theta(t)$ is $x(t) = \frac{1}{4}t^2 + 1$.

The model observation data are constructed by adding Gaussian white noise with zero mean to the true solution. The observation points are prepared within the domain of $[0, 1]$ with sampling time $T_s = 5$ ms (i.e. 201 data points). The obtained results are shown in Fig. 4.12. The root mean square errors (RMSE) for different values of signal to noise ratio are given in Table 4.7. The results reveal that higher order accuracy can be achieved by increasing the value of signal to noise ratio.

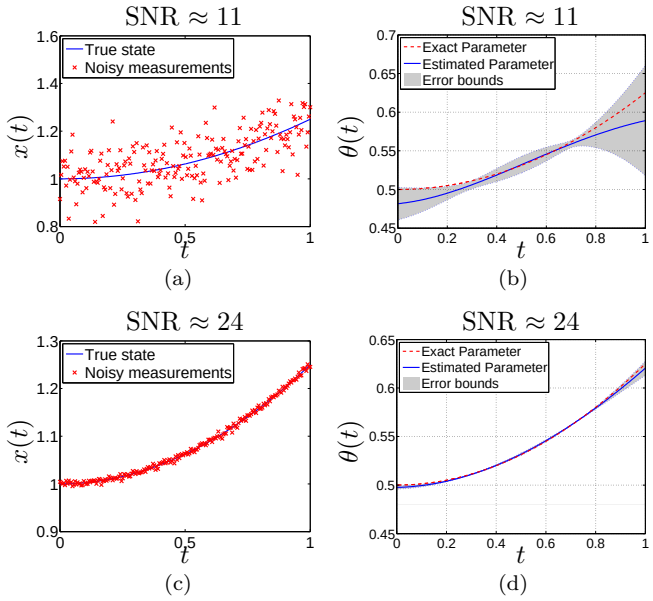


Figure 4.12: Estimation of time varying parameter $\theta(t)$ in Problem 11 from observational data for different values of signal to noise ratio.

4.7 Conclusions

In this chapter, a novel approach based on LSSVM is proposed for estimation of the unknown parameters of a dynamical system governed by ODEs and DDEs. The ability of LSSVM for producing a continuous output is exploited to generate a closed form solution for the time-varying parameters of the model understudy. The approach showed a comparable performance compared to other existing methods and can provide a good initial estimate for the parameters that can be used as a starting point for conventional optimization approaches that seek the optimal model parameters.

Chapter 5

Non-Parallel Classifiers

In this chapter, a general framework of non-parallel support vector machines, which involves a regularization term, a scatter loss and a misclassification loss is introduced. When dealing with binary problems, the framework with proper losses covers some existing non-parallel classifiers, such as multisurface proximal support vector machine via generalized eigenvalues, twin support vector machines, and their least squares version. The possibility of incorporating different existing scatter and misclassification loss functions into the general framework is discussed. Moreover, in contrast with the mentioned methods, which apply kernel-generated surfaces, we directly apply the kernel trick in the dual and then obtain nonparametric models. Therefore, one does not need to formulate two different primal problems for the linear and nonlinear kernel respectively. In addition, experimental results are given to illustrate the performance of different loss functions. In addition the possibility of learning from few labeled and large amount of unlabeled data points using a non-parallel classifier is discussed through introducing the NP-Semi-KSC.

5.1 Related Work

For binary classification problems, both SVMs and LSSVMs aim at constructing two parallel hyperplanes (or the hyperplanes in the feature space) to do classification. An extension is to consider non-parallel hyperplanes. The concept of applying two non-parallel hyperplanes was first introduced in [103], where two non-parallel hyperplanes were determined via solving two generalized eigenvalue problems, and called generalized eigenvalue proximal

SVM (GEPSVM). In this case one obtains two non-parallel hyperplanes where each one is as close as possible to the data points of one class and as far as possible from the data points of the other class. Recently many approaches, based on non-parallel hyperplanes, have been developed for classification, regression and feature selection tasks (see [184, 132, 133, 144]).

The authors in [83] modified GEPSVM and proposed a non-parallel classifier called Twin Support Vector Machines (TWSVM), that obtains two non-parallel hyperplanes by solving a pair of quadratic programming problems. An improved TWSVM termed as TBSVM is given in [145] where the structural risk is minimized. Motivated by the ideas given in [157] and [61], recently least squares twin support vector machines (LSTSVM) is presented in [8], where the primal quadratic problems of TSVM is modified into least squares problem via replacing inequalities constraints by equalities.

In the above mentioned approaches, kernel-generated surfaces are used for designing a nonlinear classifier. In addition one has to construct different primal problems depending on whether a linear or nonlinear kernel is applied. It is the purpose of this chapter to formulate a non-parallel support vector machine classifier for which we can directly apply the kernel trick and thus it enjoys the primal and dual properties as in classical support vector machines classifiers. A general framework of non-parallel support vector machine, which consists of a regularization term, a scatter loss and a misclassification loss, is provided. The framework is designed for multi-class problems. Several choices for the losses are investigated. The corresponding nonparametric models are given via considering the dual problems and the kernel trick.

5.2 Non-parallel Support Vector Machine

Let us consider a given training dataset $\{x_i, y_i\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$, y_i is the label of the i-th data point and there are M classes. Here the one-vs-all strategy is utilized to build the codebook, i.e., the training points belonging to the m-th class are labeled by $+1$ and all the remaining data from the rest of the classes are considered to have negative labels. The index set corresponding to class m is denoted by $\mathcal{I}_m$. We seek non-parallel hyperplanes in the feature space:

$$f_m(x) = w_m^T \varphi_m(x) + b_m = 0, \quad m = 1, 2, \ldots, M$$

each of which is as close as possible to the points of its own class and as far as possible from the data points of the other class.

5.2.1 General formulation

In the primal, the hyperplane $f_m(x) = 0$ for class m can be constructed by the following problem, [112]:

$$\begin{aligned} \min_{w_m, b_m, e, \xi} \quad & \frac{1}{2} w_m^T w_m + \frac{\gamma_1}{2} \sum_{i \in \mathcal{I}_m} L_{(1)}(e_i) + \frac{\gamma_2}{2} \sum_{i \notin \mathcal{I}_m} L_{(2)}(\xi_i) \\ \text{subject to} \quad & w_m^T \phi_m(x_i) + b_m = e_i, \forall i \in \mathcal{I}_m \\ & 1 + \left(w_m^T \phi_m(x_i) + b_m \right) = \xi_i, \forall i \notin \mathcal{I}_m. \end{aligned} \tag{5.1}$$

After solving (5.1) for $m = 1, 2, \ldots, M$, we obtain M non-parallel hyperplanes in the feature space. Then the label of the new test point x^* is determined depending on the perpendicular distances of the test point from the hyperplanes. Mathematically, the decision rule can be written as follows:

$$\widehat{\text{Label}}(x^*) = \arg \min_{m=1,2,\ldots,M} \{d_m(x^*)\}, \tag{5.2}$$

where the perpendicular distance $d_m(x^*)$ is calculated by

$$d_m(x^*) = \frac{\left| w_m^T \phi_m(x^*) + b_m \right|}{\|w_m\|_2}, \quad m = 1, 2, \ldots, M.$$

The target of (5.1) is to establish a hyperplane which is close to the points in class $\mathcal{I}_m$ and also is far away from the points that are not in this class. Therefore, any scatter loss function can be used for $L_{(1)}(\cdot)$ and at the same time any misclassification loss function can be utilized for $L_{(2)}(\cdot)$. Possible choices for $L_{(1)}(\cdot)$ include least squares, ϵ-insensitive tube, absolute, and Huber loss. For $L_{(2)}(\cdot)$, one can consider least squares, hinge, or squared hinge loss. Different loss has its own statistical properties and is suitable for different tasks. The proposed general formulation (5.1) is to handle multi-class problems, for which we essentially solve a series of binary problems. In the binary problem related to class m, we regard $x_i, i \in \mathcal{I}_m$ and the remaining points as two classes. Hence, the basic scheme of (5.1) for multi-class problems and binary problems is similar. For the convenience of expression, we focus on binary problems in theoretical discussion and evaluate multi-class problems in numerical experiments. Besides, for each class, one can apply a different nonlinear feature mapping in (5.1). Here we discuss the case that a unique $\phi(x)$ is used for all the classes.

Remark 5.2.1. *In general if one uses a nonlinear feature map $\varphi(\cdot)$, obviously two non-parallel hyper-surfaces will be obtained. Here the term "hyperplane" is used.*

5.2.2 Related existing methods

For a binary problem, we assume that there are n_1 points in class 1 and n_2 points in class 2, i.e., there are n_1 elements in $\mathcal{I}_1$ and n_2 in $\mathcal{I}_2$. Suppose X_1 and X_2 are matrices, containing data points of class 1 and 2 respectively. The corresponding matrices with feature mapping $\varphi(\cdot)$ are denoted by Φ_1 and Φ_2, i.e. the i-th row of Φ_1 is the vector $\varphi(x_i), i \in \mathcal{I}_1$, and so is Φ_2. Denote $Y_{n_1} = \text{diag}\{+1\}_{i=1}^{n_1} \in \mathbb{R}^{n_1 \times n_1}$, $Y_{n_2} = \text{diag}\{-1\}_{i=1}^{n_2} \in \mathbb{R}^{n_2 \times n_2}$, and 1_n as an n dimensional vector with all components equal to one. Then the non-parallel SVM (5.1) can be written in matrix formulation as the following two problems:

$$\begin{aligned} \min_{w_1,b_1,e,\xi} \quad & \frac{1}{2} w_1^T w_1 + \frac{\gamma_1}{2} L_{(1)}(e) + \frac{\gamma_2}{2} L_{(2)}(\xi) \\ \text{subject to} \quad & \Phi_1 w_1 + b_1 1_{n_1} = e \\ & Y_{n_2} \left[\Phi_2 w_1 + b_1 1_{n_2} \right] + \xi = 1_{n_2}, \end{aligned} \tag{5.3}$$

and

$$\begin{aligned} \min_{w_2,b_2,e,\xi} \quad & \frac{1}{2} w_2^T w_2 + \frac{\gamma_1}{2} L_{(1)}(e) + \frac{\gamma_2}{2} L_{(2)}(\xi) \\ \text{subject to} \quad & \Phi_2 w_2 + b_2 1_{n_2} = e \\ & Y_{n_1} \left[\Phi_1 w_2 + b_2 1_{n_1} \right] + \xi = 1_{n_1}. \end{aligned} \tag{5.4}$$

As discussed previously, $L_{(1)}(\cdot)$ could be any scatter loss function and any misclassification loss can be used in $L_{(2)}(\cdot)$. For example, if one chooses least squares loss for $L_{(1)}(\cdot)$ and hinge loss for $L_{(2)}(\cdot)$ and let $\gamma_1 = \gamma_2 \to \infty$, the problem formulations (5.28) and (5.29), when a linear kernel is used, will reduce to TWSVM introduced in [83]:

$$\begin{aligned} \text{TWSVM1} \qquad \min_{w_1,b_1,\xi} \quad & \frac{1}{2} \| X_1 w_1 + b_1 1_{n_1} \|^2 + C_1 1_{n_2}^T \xi \\ \text{subject to} \quad & -(X_2 w_1 + b_1 1_{n_2}) + \xi \geq 1_{n_2}, \end{aligned} \tag{5.5}$$

$$\begin{aligned} \text{TWSVM2} \qquad \min_{w_2,b_2,\xi} \quad & \frac{1}{2} \| X_2 w_2 + b_2 1_{n_2} \|^2 + C_2 1_{n_1}^T \xi \\ \text{subject to} \quad & (X_1 w_2 + b_2 1_{n_1}) + \xi \geq 1_{n_1}. \end{aligned} \tag{5.6}$$

Another example is choosing the least squares loss for both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$. Again, letting $\gamma_1 = \gamma_2 \to \infty$ in (5.28) and (5.29) and using a linear kernel, one obtains the LSTSVM formulation reported in [8]

$$\begin{aligned} \text{LSTSVM1} \qquad & \min_{w_1,b_1,\xi} \quad \frac{1}{2}\|X_1 w_1 + b_1 1_{n_1}\|^2 + \frac{C_1}{2}\xi^T\xi \\ & \text{subject to} \quad -(X_2 w_1 + b_1 1_{n_2}) + \xi = 1_{n_2}, \end{aligned} \tag{5.7}$$

$$\begin{aligned} \text{LSTSVM2} \qquad & \min_{w_2,b_2,\xi} \quad \frac{1}{2}\|X_2 w_2 + b_2 1_{n_2}\|^2 + \frac{C_2}{2}\xi^T\xi \\ & \text{subject to} \quad (X_1 w_2 + b_2 1_{n_1}) + \xi = 1_{n_1}. \end{aligned} \tag{5.8}$$

In contrast with the classical support vector machines technique, TWSVM and LSTSVM do not take the structural risk minimization into account. For TWSVM, the authors in [145] gave an improvement by adding a regularization term in the objective function aiming at minimizing the structural risk by maximizing the margin. This method is called TBSVM, where the bias term is also penalized. But penalizing the bias term will not affect the result significantly and will change the optimization problem slightly. From a geometric point of view it is sufficient to penalize the norm of w in order to maximize the margin.

Another noticeable point is that TWSVM, LSTSVM, and TBSVM use a kernel generated surface to apply nonlinear kernels. As opposed to these methods, in our formulation, the burden of designing another two optimization formulations, when a nonlinear kernel is used, is reduced by applying Mercer's theorem and the kernel trick directly, which will be investigated in the following section.

5.3 Different Loss Functions

There are several possibilities for choosing the loss functions $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$. Our target is to make the points in one class clustered in the hyperplane by minimizing $L_{(1)}(\cdot)$, which hence should be a scatter loss. For this aim, we prefer to use the least squares loss for $L_{(1)}(\cdot)$, because the related problem is easy to handle. Its weak point is that the least squares loss is sensitive to large outliers, then one may also consider ℓ_1-norm or Huber loss under the proposed framework. For $L_{(2)}(\cdot)$, which penalizes misclassification error to push the points in other classes away from the hyperplane, we need a misclassification loss. In what follows, we illustrate the following loss functions used in (5.3) and (5.4). Other loss functions can be discussed similarly:

- Least squares loss for $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$ (will be referred to as LS-LS case).
- Least squares loss for $L_{(1)}(\cdot)$ and hinge loss for $L_{(2)}(\cdot)$ (will be referred to as LS-Hinge case).
- Least squares loss for $L_{(1)}(\cdot)$ and pinball loss for $L_{(2)}(\cdot)$ (will be referred to as LS-Pinball case).

The above-mentioned loss functions are depicted in Fig 5.1.

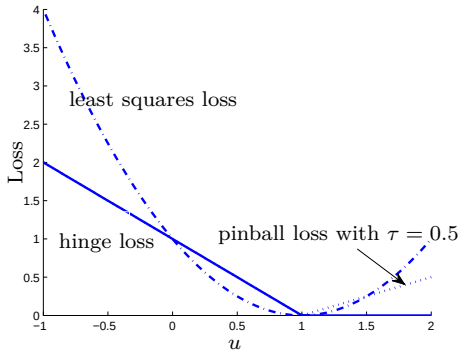


Figure 5.1: Some loss functions for $L_{(2)}(\cdot)$: hinge loss (solid line), least squares loss (dot-dashed line), and pinball loss with $\tau = 0.5$ (dotted line).

5.3.1 Case: LS-LS loss

We first investigate the case using a least squares loss in both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$. Due to the fact that applying least squares loss will lead to a set of linear systems, this choice has much lower computational cost in comparison with other loss functions, which may result in solving quadratic programming problems or nonlinear systems of equations. Specifically, using least squares loss in (5.28) and (5.29) leads to the following problems:

$$\begin{aligned} \min_{w_1,b_1,e,\xi} \quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}e^T e + \frac{\gamma_2}{2}\xi^T \xi \\ \text{subject to} \quad & \Phi_1 w_1 + b_1 1_{n_1} = e \\ & Y_{n_2}\Big[\Phi_2 w_1 + b_1 1_{n_2}\Big] + \xi = 1_{n_2}, \end{aligned} \tag{5.9}$$

and

$$\begin{aligned} \min_{w_2,b_2,e,\xi} \quad & \frac{1}{2}w_2^T w_2 + \frac{\gamma_1}{2}e^T e + \frac{\gamma_2}{2}\xi^T \xi \\ \text{subject to} \quad & \Phi_2 w_2 + b_2 1_{n_2} = e \\ & Y_{n_1}\left[\Phi_1 w_2 + b_2 1_{n_1}\right] + \xi = 1_{n_1}. \end{aligned} \tag{5.10}$$

In this case, problems (5.9) or (5.10) become a quadratic minimizations under linear equality constraints, which enables a straightforward solution.

The obtained formulations (5.9) and (5.10) are closely related to LSTSVM (5.7) and (5.8). An important difference is that there are regularization terms involved in (5.9) and (5.10), which makes the kernel trick applicable to obtain nonparametric models. In [8], the kernel generated surfaces were introduced to LSTSVM, which does not consider structural risk minimization and also brings the burden of designing another two optimization formulations when a nonlinear kernel is used. Our nonparametric model can be directly obtained from the dual problem of (5.9) and (5.10), illustrated below.

Theorem 5.3.1. *Given a positive definite kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $K(t,s) = \varphi(t)^T\varphi(s)$ and regularization constants $\gamma_1, \gamma_2 \in \mathbb{R}^+$, the dual problem of (5.9) is posed as:*

$$\left[\begin{array}{c|c|c} \Omega_{11} + I_{n_1}/\gamma_1 & \Omega_{12}Y_{n_2} & 1_{n_1} \\ \hline Y_{n_2}\Omega_{21} & \Omega_{22} + I_{n_2}/\gamma_2 & Y_{n_2}1_{n_2} \\ \hline 1_{n_1}^T & 1_{n_2}^T Y_{n_2} & 0 \end{array}\right] \left[\begin{array}{c} \alpha_1 \\ \hline \beta_1 \\ \hline b_1 \end{array}\right] = \left[\begin{array}{c} 0_{n_1} \\ \hline 1_{n_2} \\ \hline 0 \end{array}\right] \tag{5.11}$$

with $\alpha_1 \in \mathbb{R}^{n_1}, \beta_1 \in \mathbb{R}^{n_2}, \Omega_{11} = \Phi_1\Phi_1^T, \Omega_{22} = \Phi_2\Phi_2^T, \Omega_{12} = \Phi_1\Phi_2^T$ and $\Omega_{21} = \Omega_{12}^T$. In other words, the elements of Ω_{11} are calculated by $K(x_i, x_j), i, j \in \mathcal{I}_1$, and so are Ω_{12}, Ω_{21} and Ω_{22}.

Proof. The Lagrangian of the constrained optimization problem (5.9) becomes

$$\begin{aligned} & \mathcal{L}(w_1, b_1, e, \xi, \alpha_1, \beta_1) = \\ & \frac{1}{2}\, w_1^T w_1 + \frac{\gamma_1}{2}e^T e + \frac{\gamma_1}{2}\xi^T\xi - \alpha_1^T\Big(\Phi_1 w_1 + b_1 1_{n_1} - e\Big) - \\ & \beta_1^T\left(Y_{n_2}\left[\Phi_2 w_1 + b_1 1_{n_2}\right] + \xi - 1_{n_2}\right) \end{aligned}$$

where α_1 and β_1 are the Lagrange multipliers corresponding to the constraints in (5.9). Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as

follows,

$$\frac{\partial \mathcal{L}}{\partial w_1} = 0 \rightarrow \; w_1 = \Phi_1^T \alpha_1 + \Phi_2^T Y_{n_2} \beta_1,$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = 0 \rightarrow \; 1_{n_1}^T \alpha_1 + 1_{n_2}^T Y_{n_2} \beta_1 = 0,$$

$$\frac{\partial \mathcal{L}}{\partial e} = 0 \rightarrow \; e = -\frac{\alpha_1}{\gamma_1},$$

$$\frac{\partial \mathcal{L}}{\partial \xi} = 0 \rightarrow \; \xi = \frac{\beta_1}{\gamma_2},$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_1} = 0 \rightarrow \; \Phi_1 w_1 + b_1 1_{n_1} - e = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \beta_1} = 0 \rightarrow \; Y_{n_2} \Big[\Phi_2 w_1 + b_1 1_{n_2} \Big] + \xi = 1_{n_2}.$$

After elimination of the primal variables w_1, e, ξ and making use of Mercer's Theorem, one can obtain the solution in the dual by solving linear system (5.11). □

Using a similar argument, one can show that the solution of optimization problem (5.4) can be obtained in the dual by solving the following linear system:

$$\left[\begin{array}{c|c|c} \Omega_{22} + I_{n_2}/\gamma_1 & \Omega_{21} Y_{n_1} & 1_{n_2} \\ \hline Y_{n_1} \Omega_{12} & \Omega_{11} + I_{n_1}/\gamma_2 & Y_{n_1} 1_{n_1} \\ \hline 1_{n_2}^T & 1_{n_1}^T Y_{n_1} & 0 \end{array}\right] \left[\begin{array}{c} \alpha_2 \\ \hline \beta_2 \\ \hline b_2 \end{array}\right] = \left[\begin{array}{c} 0_{n_2} \\ \hline 1_{n_1} \\ \hline 0 \end{array}\right] \tag{5.12}$$

with $\alpha_2 \in \mathbb{R}^{n_2}, \beta_2 \in \mathbb{R}^{n_1}$.

Via solving (5.11) and (5.12), we obtain the optimal dual variables $\alpha_{1,2}$, $\beta_{1,2}$, and $b_{1,2}$. Then for the unseen test data points $\mathcal{D}^{\text{test}} = \{x_j^*\}_{j=1}^{n_{\text{test}}}$ the labels can be determined using (5.2) where

$$\begin{aligned} d_1(\mathcal{D}^{\text{test}}) =& \frac{|\Phi_{\text{test}} w_1 + b_1 1_{n_{\text{test}}}|}{\|w_1\|_2} \\ =& \frac{|\Phi_{\text{test}} (\Phi_1^T \alpha_1 + \Phi_2^T Y_{n_2} \beta_1) + b_1 1_{n_{\text{test}}}|}{\|\Phi_1^T \alpha_1 + \Phi_2^T Y_{n_2} \beta_1\|_2}, \end{aligned}$$

and

$$d_2(\mathcal{D}^{\text{test}}) = \frac{|\Phi_{\text{test}} w_2 + b_2 1_{n_{\text{test}}}|}{\|w_2\|_2}$$

$$= \frac{|\Phi_{\text{test}}(\Phi_2^T \alpha_2 + \Phi_1^T Y_{n_1} \beta_2) + b_2 1_{n_{\text{test}}}|}{\|\Phi_2^T \alpha_2 + \Phi_1^T Y_{n_1} \beta_2\|_2}.$$

Here $\Phi_{\text{test}} = [\varphi(x_1^*), \ldots, \varphi(x_{n_{\text{test}}}^*)]^T$. Thanks to the KKT optimality conditions, w_1 and w_2 are written in terms of Lagrange multipliers.

Next we will show that when we set $\gamma_1 = \gamma_2$, (5.11) and (5.12) reduce to least squares support vector machine classifier [157], given below,

$$\begin{aligned} \min_{w,b,e} \quad & \frac{1}{2} w^T w + \frac{\gamma}{2} e^T e \\ \text{subject to} \quad & Y \Big[\Phi w + b 1_N \Big] = 1_N - e, \end{aligned} \tag{5.13}$$

where $N = n_1 + n_2$ is the number of training data in both class 1 and class 2. and Y is a diagonal matrix which contains the labels of all classes i.e. -1 and +1 on the main diagonal.

Theorem 5.3.2. *Problems (5.9) and (5.10) are equivalent to the standard least squares support vector machine classifier (5.13) when $\gamma_1 = \gamma_2$.*

Proof. Consider problem (5.9) with least squares loss and $\gamma_1 = \gamma_2$. We introduce a new variable $\tilde{b}_1 = b_1 + 1/2$, and rewrite (5.9) as follows:

$$\begin{aligned} \min_{w_1, \tilde{b}_1, e, \xi} \quad & \frac{1}{2} w_1^T w_1 + \frac{\gamma_1}{2} e^T e + \frac{\gamma_2}{2} \xi^T \xi \\ \text{subject to} \quad & Y_{n_1} \Big[\Phi_1 w_1 + \tilde{b}_1 1_{n_1} \Big] - e = \frac{1}{2} 1_{n_1} \\ & Y_{n_2} \Big[\Phi_2 w_1 + \tilde{b}_1 1_{n_2} \Big] + \xi = \frac{1}{2} 1_{n_2}, \end{aligned} \tag{5.14}$$

where Y_{n_1} is defined as previously. Since $\gamma_1 = \gamma_2$, by combining the constraints, one can rewrite (5.14) as follows:

$$\begin{aligned} \min_{w_1, \tilde{b}_1, \tilde{e}} \quad & \frac{1}{2} w_1^T w_1 + \frac{\gamma}{2} \tilde{e}^T \tilde{e} \\ \text{subject to} \quad & \tilde{e} = \frac{1}{2} 1_N - Y_N \Big[\Phi 2 w_1 + 2 \tilde{b}_1 1_N \Big], \end{aligned} \tag{5.15}$$

where $\Phi = \begin{bmatrix} \Phi_1 \\ \Phi_2 \end{bmatrix}, \tilde{e} = \begin{bmatrix} e \\ \xi \end{bmatrix}, Y_N = \begin{bmatrix} Y_{n_1} \\ Y_{n_2} \end{bmatrix}$ and $1_N = \begin{bmatrix} 1_{n_1} \\ 1_{n_2} \end{bmatrix}$.

Now let $\bar{w} = 2w_1$ and $\bar{b} = 2\tilde{b}$, then one can find that (5.15) is equivalent to the following optimization problem:

$$\begin{aligned} \min_{\bar{w},\bar{b},\bar{e}} \quad & \frac{1}{2}(\bar{w}\,)^T(\bar{w}\,) + \frac{\gamma}{2}(\bar{e}\,)^T(\bar{e}\,) \\ \text{subject to} \quad & \bar{e} = 1_N - Y_N\left[\Phi\bar{w} + \bar{b}1_N\right], \end{aligned} \tag{5.16}$$

which is indeed the classical LSSVM classifier formulation. □

Similarly one can demonstrate that (5.29) with least squares loss and $\gamma_1 = \gamma_2$ will be equivalent to (5.13). This relationship implies that the LS-LS is an extension to LSSVM, from which we can start from LSSVM and then improve the classifier using LS-LS model.

5.3.2 Case: LS-Hinge loss

In the non-parallel SVM framework (5.3) and (5.4), if we choose the least squares loss for $L_{(1)}(\cdot)$ and hinge loss for $L_{(2)}(\cdot)$, then the problem in the primal has the following form

$$\begin{aligned} \min_{w_1,b_1,e,\xi} \quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}e^T e + \gamma_2 1_{n_2}^T \xi \\ \text{subject to} \quad & \Phi_1 w_1 + b_1 1_{n_1} = e \\ & Y_{n_2}\left[\Phi_2 w_1 + b_1 1_{n_2}\right] + \xi \geq 1_{n_2} \\ & \xi \geq 0_{n_2}, \end{aligned} \tag{5.17}$$

and

$$\begin{aligned} \min_{w_2,b_2,e,\xi} \quad & \frac{1}{2}w_2^T w_2 + \frac{\gamma_1}{2}e^T e + \gamma_2 1_{n_1}^T \xi \\ \text{subject to} \quad & \Phi_2 w_2 + b_2 1_{n_2} = e \\ & Y_{n_1}\left[\Phi_1 w_2 + b_2 1_{n_1}\right] + \xi \geq 1_{n_1} \\ & \xi \geq 0_{n_1}. \end{aligned} \tag{5.18}$$

The Lagrangian of the constrained optimization problem (5.17) becomes

$$\mathcal{L}(w_1, b_1, e, \xi, \alpha_1, \beta_1, \beta_2) = \tag{5.19}$$

$$\frac{1}{2} w_1^T w_1 + \frac{\gamma_1}{2} e^T e + \gamma_2 1_{n_2}^T \xi - \alpha_1^T \Big(\Phi_1 w_1 + b_1 1_{n_1} - e \Big) - \tag{5.20}$$

$$\beta_1^T \Big(Y_{n_2} \Big[\Phi_2 w_1 + b_1 1_{n_2} \Big] + \xi - 1_{n_2} \Big) - \beta_2^T \xi, \tag{5.21}$$

where α_1 and $\beta_1 \geq 0$ and $\beta_2 \geq 0$ are the Lagrange multipliers corresponding to the constraints in (5.17). Then the solution is characterized by the saddle point of the Lagrangian

$$\max_{\alpha_1, \beta_1, \beta_2} \quad \min_{w_1, b_1, e, \xi} \mathcal{L}(w_1, b_1, e, \xi, \alpha_1, \beta_1, \beta_2).$$

The Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\frac{\partial \mathcal{L}}{\partial w_1} = 0 \rightarrow \; w_1 = \Phi_1^T \alpha_1 + \Phi_2^T Y_{n_2} \beta_1,$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = 0 \rightarrow \; 1_{n_1}^T \alpha_1 + 1_{n_2}^T Y_{n_2} \beta_1 = 0,$$

$$\frac{\partial \mathcal{L}}{\partial e} = 0 \rightarrow \; e = -\frac{\alpha_1}{\gamma_1},$$

$$\frac{\partial \mathcal{L}}{\partial \xi} = 0 \rightarrow \; 0 \leq \beta_1 \leq \gamma_2 1_{n_2}$$

$$\beta_1 \geq 0$$

$$\beta_2 \geq 0.$$

Substituting the equations obtained from KKT conditions into the Lagrangian (5.19) and maximizing with respect to the Lagrange multipliers yields the following dual problem of (5.17):

$$\begin{aligned} \max_{\mu_1} \quad & -\frac{1}{2} \mu_1^T H_1 \mu_1 + F_1 \mu_1 \\ \text{subject to} \quad & A_1 \mu_1 = 0 \\ & 0 \leq \beta_1 \leq \gamma_2 1_{n_2}, \end{aligned} \tag{5.22}$$

where $H_1 = \left[\begin{array}{c|c} \Omega_{11} + \gamma_1^{-1} I_{n_1} & \Omega_{12} Y_{n_2} \\ \hline Y_{n_2} \Omega_{21} & Y_{n_2} \Omega_{22} Y_{n_2} \end{array}\right]$, $\mu_1 = [{\alpha_1}^T, {\beta_1}^T]^T$, $F_1 = [0_{n_1}^T, 1_{n_2}^T]$, and $A_1 = [1_{n_1}^T, 1_{n_2}^T Y_{n_2}]$.

Correspondingly, the dual problem of (5.18) can be constructed as follows:

$$\begin{array}{ll} \max\limits_{\mu_2} & -\dfrac{1}{2}\mu_2^T H_2 \mu_2 + F_2 \mu_2 \\ \text{subject to} & A_2 \mu_2 = 0 \\ & 0 \leq \beta_1 \leq \gamma_2 1_{n_1}, \end{array} \tag{5.23}$$

where $H_2 = \left[\begin{array}{c|c} \Omega_{22} + \gamma_1^{-1} I_{n_2} & \Omega_{21} Y_{n_1} \\ \hline Y_{n_1} \Omega_{12} & Y_{n_1} \Omega_{11} Y_{n_1} \end{array}\right]$, $\mu_2 = [{\alpha_1}^T, {\beta_1}^T]^T$, $F_2 = [0_{n_2}^T, 1_{n_1}^T]$, and $A_2 = [1_{n_2}^T, 1_{n_1}^T Y_{n_1}]$.

In (5.22) and (5.23) the kernel generated surfaces are not used and our formulation enjoys the advantages of having primal and dual formulations with applying the kernel trick. Also the structural risk minimization is obtained by means of the regularization terms $w_1^T w_1$ and $w_2^T w_2$.

If one uses the least squares loss for both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$, then in the dual a set of linear systems have to be solved but no sparsity will be achieved. Whereas if one chooses typical SVM losses, e.g., ϵ-insensitive zone loss for $L_{(1)}(\cdot)$, and hinge loss for $L_{(2)}(\cdot)$, then in the dual the hyperparameters of the model can be obtained by solving a convex quadratic optimization problem. In this case sparsity is enhanced since the training points that are correctly classified and are far enough from the margins will have no influence on the decision boundary. One can also use Huber loss function for $L_{(1)}(\cdot)$ to cope with the noise or outliers in the data set.

5.3.3 Case: LS-Pinball loss

When the hinge loss is minimized, the distance that we maximize is related to the nearest points which is prone to be sensitive to noise. Therefore attempts have been made to overcome this weak point by changing the definition of the distance between two sets. For instance, if one uses the distance of the nearest 20% points to measure the distance between two sets, the result is more robust. Such distance is a kind of quantile value, which is closely related to pinball loss [84, 150]. In classification, we consider the following definition of pinball loss:

$$L_\tau(u) = \begin{cases} u, & u \geq 0, \\ -\tau u, & u < 0. \end{cases}$$

The pinball loss has been used for classification problems in [76]. The advantage of using the pinball loss holds as well for non-parallel classifiers. The corresponding model can be formulated as the following quadratic programming problems,

$$\begin{aligned} \min_{w_1,b_1,e,\xi} \quad & \frac{1}{2}{w_1}^T w_1 + \frac{\gamma_1}{2} e^T e + \gamma_2 1_{n_2}^T \xi \\ \text{subject to} \quad & \Phi_1 w_1 + b_1 1_{n_1} = e \\ & Y_{n_2}\left[\Phi_2 w_1 + b_1 1_{n_2}\right] + \xi \geq 1_{n_2} \\ & Y_{n_2}\left[\Phi_2 w_1 + b_1 1_{n_2}\right] - \frac{1}{\tau}\xi \leq 1_{n_2}, \end{aligned} \tag{5.24}$$

and

$$\begin{aligned} \min_{w_2,b_2,e,\xi} \quad & \frac{1}{2}{w_2}^T w_2 + \frac{\gamma_1}{2} e^T e + \gamma_2 1_{n_1}^T \xi \\ \text{subject to} \quad & \Phi_2 w_2 + b_2 1_{n_2} = e \\ & Y_{n_1}\left[\Phi_1 w_2 + b_2 1_{n_1}\right] + \xi \geq 1_{n_1} \\ & Y_{n_1}\left[\Phi_1 w_2 + b_2 1_{n_1}\right] - \frac{1}{\tau}\xi \leq 1_{n_1}. \end{aligned} \tag{5.25}$$

Similar to the previous discussions (see subsection 5.3.2), we can derive the corresponding nonparametric model. The dual problem of (5.24) is

$$\begin{aligned} \max_{\mu_1} \quad & -\frac{1}{2}\mu_1^T H_1 \mu_1 + F_1 \mu_1 \\ \text{subject to} \quad & A_1 \mu_1 = 0 \\ & -\tau\gamma_2 1_{n_2} \leq \beta_1 \leq \gamma_2 1_{n_2}, \end{aligned} \tag{5.26}$$

and that of (5.25) is

$$\begin{aligned} \max_{\mu_2} \quad & -\frac{1}{2}\mu_2^T H_2 \mu_2 + F_2 \mu_2 \\ \text{subject to} \quad & A_2 \mu_2 = 0 \\ & -\tau\gamma_2 1_{n_1} \leq \beta_1 \leq \gamma_2 1_{n_1}. \end{aligned} \tag{5.27}$$

When $\tau = 0$, (5.26) and (5.27) reduces to (5.22) and (5.23), respectively. From this point of view, the LS-Pinball is an extension to the LS-Hinge. This

relationship also can be observed via comparing the hinge loss and pinball loss in the primal.

Theorem 5.3.2 tells us that LS-LS with particular parameters reduces to LSSVM. We are also interested in the relationship between other non-parallel classifiers and parallel ones. In parallel classification methods, only one loss function is minimized. In the proposed non-parallel framework (5.1), there are two loss functions involved. Only when we choose a unique loss for both $L_{(1)}(\cdot)$ and $L_{(2)}(\cdot)$, it is possible to reduce the non-parallel models to parallel ones.

5.4 Guidelines for the User

The proposed framework for constructing the non-parallel classifier consists of two types of loss functions: scatter and misclassification. As mentioned previously, any scatter loss function can be used for $L_{(1)}(\cdot)$ and at the same time any misclassification loss can be employed for $L_{(2)}(\cdot)$. Depending on the prior knowledge about the data under study, one may choose a specific scatter or misclassification loss function. For instance if the data is corrupted by label noise one may prefer to use the hinge or pinball loss misclassification which are less sensitive to outliers compared to least squares loss. In case no prior knowledge is available, then, in general, choosing the loss functions can be regarded as user defined choice. One may try different loss functions and select the one with minimum misclassification error on the validation set. Based on the statistical properties of each of the loss functions the following qualitative conclusion can be drawn.

Table 5.1: Qualitative conclusion for different loss functions

Type of noise	LS-LS	LS-Hinge	LS-Pinball
Label noise	✗	✓	✓
Feature noise	✓	✗	✓

Remark 5.4.1. *One may notice that according to Theorem 5.3.2 the LSSVM is a special case of LS-LS (with the ratio $r = 1$). Therefore in practice one can start with the LSSVM algorithm and gradually change (tune) the ratio r, to obtain the non-parallel classifier with a better performance compared to the LSSVM. After reaching the stage where the non-parallel classifier is built, one then can choose empirically the loss function that obtains the minimum misclassification error on the validation set.*

Algorithm 6: Guidelines for the user

Input: Training data set $\mathcal{D} = \{x_i\}_{i=1}^N$, labels $\{y_i\}_{i=1}^N$, the tuning parameters (if any)

Output: Class membership of test data points $\mathcal{D}^{test}$

Option 1. Try all combinations of the loss functions and choose the one with minimum misclassification error on the validation set.
Option 2. Start with the LSSVM approach,
Employ Theorem 5.3.2 and obtain a non-parallel classifier,
Search for the best possible loss functions with minimum misclassification error on the validation set.

5.5 Non-parallel Semi-Supervised KSC

Here the LS-LS loss function introduced in section 5.3 is used to build a non-parallel semi-supervised classifier (NP-Semi-KSC). The model uses the Kernel Spectral Clustering as core model and the available labeled data points are integrated into the primal optimization problem via a regularization term. In the training phase, the algorithm learns two non-parallel hyperplanes using both labeled and unlabeled data points. Suppose the training data set $\mathcal{X}$ consists of M data points and is defined as follows:

$$\mathcal{X} = \{\underbrace{x_1, ..., x_N}_{\substack{Unlabeled \\ (\mathcal{X}_U)}}, \underbrace{x_{N+1}, .., x_{N+\ell_1}}_{\substack{Labeled\,with\,(+1) \\ (\mathcal{X}_{L_1})}}, \underbrace{x_{N+\ell_1+1}, .., x_{N+\ell_1+\ell_2}}_{\substack{Labeled\,with\,(-1) \\ (\mathcal{X}_{L_2})}}\}$$

where $\{x_i\}_{i=1}^M \in \mathbb{R}^d$. Let us decompose the training data into unlabeled and labeled parts as $\mathcal{X} = \mathcal{X}_U \cup \mathcal{X}_{L_1} \cup \mathcal{X}_{L_2}$ where subsets $\mathcal{X}_U$, $\mathcal{X}_{L_1}$ and $\mathcal{X}_{L_2}$ consisting of N_U unlabeled samples, N_{L1} samples of class I and N_{L2} samples of II respectively. Note that the total number of samples is $M = N_U + N_{L1} + N_{L2}$. The target values are denoted by set $\mathcal{Y}$ which consists of binary labels:

$$\mathcal{Y} = \{\underbrace{+1, \ldots, +1}_{y^1}, \underbrace{-1, \ldots, -1}_{y^2}\}.$$

The same decomposition procedure is applied for the available target values i.e. $\mathcal{Y} = y^1 \cup y^2$ where y^1 and y^2 consist of labels of the samples from class I and II respectively.

We seek two non-parallel hyperplanes

$$f_1(x) = w_1^T \varphi(x) + b_1 = 0, \quad f_2(x) = w_2^T \varphi(x) + b_2 = 0$$

where each one is as close as possible to the points of its own class and as far as possible from the data of the other class.

5.5.1 Primal-Dual formulation of the method

We formulate a non-parallel semi-supervised KSC, in the primal, as the following two optimization problems [115]:

$$\begin{aligned} \min_{w_1,b_1,e,\eta,\xi} \quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}\eta^T\eta + \frac{\gamma_2}{2}\xi^T\xi - \frac{\gamma_3}{2}e^T D^{-1}e \\ \text{subject to} \quad & w_1^T\varphi(x_i) + b_1 = \eta_i,\ \forall x_i \in I \\ & y_i^2\left[w_1^T\varphi(x_i) + b_1\right] + \xi_i = 1,\ \forall x_i \in II \\ & w_1^T\varphi(x_i) + b_1 = e_i,\ \forall x_i \in \mathcal{X} \end{aligned} \tag{5.28}$$

where γ_1, γ_2 and $\gamma_3 \in \mathbb{R}^+, b_1 \in \mathbb{R}$, $\eta \in \mathbb{R}^{N_{L1}}$, $\xi \in \mathbb{R}^{N_{L2}}$, $e \in \mathbb{R}^M$, $w_1 \in \mathbb{R}^h$. $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^h$ is the feature map and h is the dimension of the feature space.

$$\begin{aligned} \min_{w_2,b_2,e,\rho,\nu} \quad & \frac{1}{2}w_2^T w_2 + \frac{\gamma_4}{2}\rho^T\rho + \frac{\gamma_5}{2}\nu^T\nu - \frac{\gamma_6}{2}e^T D^{-1}e \\ \text{subject to} \quad & w_2^T\varphi(x_i) + b_2 = \rho_i,\ \forall x_i \in II \\ & y_i^1\left[w_2^T\varphi(x_i) + b_2\right] + \nu_i = 1,\ \forall x_i \in I \\ & w_2^T\varphi(x_i) + b_2 = e_i,\ \forall x_i \in \mathcal{X} \end{aligned} \tag{5.29}$$

where γ_4, γ_5 and $\gamma_6 \in \mathbb{R}^+$. $b_2 \in \mathbb{R}$, $\rho \in \mathbb{R}^{N_{L2}}$, $\nu \in \mathbb{R}^{N_{L1}}$, $e \in \mathbb{R}^M$, $w_2 \in \mathbb{R}^h$. $\varphi(\cdot)$ is defined as previously.

The intuition for the above formulation can be expressed as follows. Consider optimization problem (5.28), the first constraint is the sum of squared distances of the points in class I from the first hyperplane i.e. $f_1(x)$ and minimizing this distance will make $f_1(x)$ to be located close to points of class I. The second constraint will push $f_1(x)$ away from data points of class II (the distance of $f_1(x)$ from the points of class II should be at least 1). The third constraint is part of the core model (KSC). A similar argument can be made for the second optimization problem (5.29). By solving optimization problems (5.28) and

(5.29) one can obtain two non-parallel hyperplanes where each one is surrounded by the data points of the corresponding cluster (class). Let us assume that class I and II consist of samples with targets (+1) and (-1) respectively. Then one can manipulate the objective function of the above optimization problems and rewrite them in primal as follows:

$$\begin{aligned} \min_{w_1,b_1,e} \quad & \frac{1}{2}w_1^T w_1 + \frac{\gamma_1}{2}e^T Ae + \frac{\gamma_2}{2}(S_1 + Be)^T(S_1 + Be) \\ & - \frac{\gamma_3}{2}e^T D^{-1}e \\ \text{subject to} \quad & w_1^T\varphi(x_i) + b_1 = e_i,\ \forall x_i \in \mathcal{X} \end{aligned} \tag{5.30}$$

$$\begin{aligned} \min_{w_2,b_2,e} \quad & \frac{1}{2}w_2^T w_2 + \frac{\gamma_4}{2}e^T Be + \frac{\gamma_5}{2}(S_2 - Ae)^T(S_2 - Ae) \\ & - \frac{\gamma_6}{2}e^T D^{-1}e \\ \text{subject to} \quad & w_2^T\varphi(x_i) + b_2 = e_i,\ \forall x_i \in \mathcal{X}, \end{aligned} \tag{5.31}$$

where

$$A = \left[\begin{array}{c|c|c} 0_{N_U\times N_U} & 0_{N_U\times N_{L1}} & 0_{N_U\times N_{L2}} \\ \hline 0_{N_{L1}\times N_U} & I_{N_{L1}} & 0_{N_{L1}\times N_{L2}} \\ \hline 0_{N_{L2}\times N_U} & 0_{N_{L2}\times N_{L1}} & 0_{N_{L2}\times N_{L2}} \end{array}\right], \tag{5.32}$$

$$B = \left[\begin{array}{c|c|c} 0_{N_U\times N_U} & 0_{N_U\times N_{L1}} & 0_{N_U\times N_{L2}} \\ \hline 0_{N_{L1}\times N_U} & 0_{N_{L1}\times N_{L1}} & 0_{N_{L1}\times N_{L2}} \\ \hline 0_{N_{L2}\times N_U} & 0_{N_{L2}\times N_{L1}} & I_{N_{L2}} \end{array}\right] \tag{5.33}$$

$$S_1 = B\,1_M,\ \ S_2 = A\,1_M. \tag{5.34}$$

Here 1_M is vector of all ones with size M. $I_{N_{L1}}$ and $I_{N_{L2}}$ are identity matrix of size $N_{L1} \times N_{L1}$ and $N_{L2} \times N_{L2}$ respectively. One can further simplify the objective of the (5.30) and (5.31) and rewrite them as follows:

$$\begin{aligned} \min_{w_1,b_1,e} \quad & \frac{1}{2}w_1^T w_1 - \frac{1}{2}e^T(\gamma_3 D^{-1} - \gamma_1 A - \gamma_2 B)e + \\ & \frac{\gamma_2}{2}(S_1^T S_1 + 2S_1^T e) \\ \text{subject to} \quad & \Phi w_1 + b_1 1_M = e, \end{aligned} \tag{5.35}$$

$$\begin{aligned} \min_{w_2,b_2,e} \quad & \frac{1}{2}w_2^T w_2 - \frac{1}{2}e^T(\gamma_6 D^{-1} - \gamma_4 B - \gamma_5 A)e + \\ & \frac{\gamma_5}{2}(S_2^T S_2 - 2S_2^T e) \\ \text{subject to} \quad & \Phi w_2 + b_2 1_M = e, \end{aligned} \tag{5.36}$$

Here $\Phi = [\varphi(x_1), \ldots, \varphi(x_M)]^T$.

Lemma 5.5.1. *Given a positive definite kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $K(x,z) = \varphi(x)^T\varphi(z)$ and regularization constants $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{R}^+$, the solution to (5.35) is obtained by solving the following dual problem [115]:*

$$(V_1 C_1 \Omega - I_M)\alpha = \gamma_2 C_1^T S_1 \tag{5.37}$$

where $V_1 = \gamma_3 D^{-1} - \gamma_1 A - \gamma_2 B$ and $C_1 = I_M - (1/1_M^T V_1 1_M)1_M 1_M^T V_1$, $\alpha = [\alpha_1, \ldots, \alpha_M]^T$ and $\Omega = \Phi\Phi^T$ is the kernel matrix.

Proof. The Lagrangian of the constrained optimization problem (5.35) becomes

$$\mathcal{L}(w_1, b_1, e, \alpha) = \frac{1}{2}w_1^T w_1 - \frac{1}{2}e^T(\gamma_3 D^{-1} - \gamma_1 A - \gamma_2 B)e +$$

$$\frac{\gamma_2}{2}(S_1^T S_1 + 2S_1^T e) + \alpha^T\Big(e - \Phi w_1 - b_1 1_M\Big)$$

where α is the vector of Lagrange multipliers. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w_1} = 0 \to\ w_1 = \Phi^T\alpha, \\ \frac{\partial \mathcal{L}}{\partial b_1} = 0 \to\ 1_M^T\alpha = 0, \\ \frac{\partial \mathcal{L}}{\partial e} = 0 \to\ \alpha = (\gamma_3 D^{-1} - \gamma_1 A - \gamma_2 B)e - \gamma_2 S_1, \\ \frac{\partial \mathcal{L}}{\partial \alpha} = 0 \to\ e = \Phi w_1 + b_1 1_M. \end{cases} \tag{5.38}$$

Elimination of the primal variables w_1, e and making use of Mercer's Theorem, will result in the following equation

$$V_1\Omega\alpha + b_1 V_1 1_M = \alpha + \gamma_2 S_1 \tag{5.39}$$

where $V_1 = \gamma_3 D^{-1} - \gamma_1 A - \gamma_2 B$. From the second KKT optimality condition and (6.5), the bias term becomes:

$$b_1 = (1/1_M^T V_1 1_M)(1_M^T \gamma_2 S_1 - 1_M^T V_1 \Omega\alpha). \tag{5.40}$$

Substituting the obtained expression for the bias term b_1 into (6.5) along with some algebraic manipulation one can obtain the solution in dual as the following linear system:

$$\gamma_2\left(I_M - \frac{V_1 1_M 1_M^T}{1_M^T V_1 1_M}\right)S_1 = V_1\left(I_M - \frac{1_M 1_M^T V_1}{1_M^T V_1 1_M}\right)\Omega\alpha - \alpha.$$

□

Lemma 5.5.2. *Given a positive definite kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $K(x, z) = \varphi(z)^T\varphi(z)$ and regularization constants $\gamma_4, \gamma_5, \gamma_6 \in \mathbb{R}^+$, the solution to (5.36) is obtained by solving the following dual problem:*

$$(I_M - V_2 C_2 \Omega)\beta = \gamma_5 C_2^T S_2, \tag{5.41}$$

where $V_2 = \gamma_6 D^{-1} - \gamma_4 B - \gamma_5 A$, $\beta = [\beta_1, \ldots, \beta_M]^T$ are the Lagrange multipliers and $C_2 = I_M - (1/1_M^T V_2 1_M)1_M 1_M^T V_2$. Ω and I_M are defined as previously.

Proof. The Lagrangian of the constrained optimization problem (5.36) becomes

$$\mathcal{L}(w_2, b_2, e, \beta) = \frac{1}{2}w_2^T w_2 - \frac{1}{2}e^T(\gamma_6 D^{-1} - \gamma_4 B - \gamma_5 A)e +$$

$$\frac{\gamma_5}{2}(S_2^T S_2 - 2S_2^T e) + \beta^T\left(e - \Phi w_2 - b_2 1_M\right)$$

where β are the Lagrange multipliers. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial w_2} = 0 \to\ w_2 = \Phi^T\beta, \\ \frac{\partial\mathcal{L}}{\partial b_2} = 0 \to\ 1_M^T\beta = 0, \\ \frac{\partial\mathcal{L}}{\partial e} = 0 \to\ \beta = (\gamma_6 D^{-1} - \gamma_4 B - \gamma_5 A)e + \gamma_5 S_2, \\ \frac{\partial\mathcal{L}}{\partial\beta} = 0 \to\ e = \Phi w_2 + b_2 1_M. \end{cases} \tag{5.42}$$

After elimination of the primal variables w_2, e and making use of Mercer's Theorem, one can obtain the following equation

$$V_2\Omega\beta + b_2 V_2 1_M = \beta - \gamma_5 S_2 \tag{5.43}$$

where $V_2 = \gamma_6 D^{-1} - \gamma_4 B - \gamma_5 A$. From (5.43) and the second KKT optimality condition, the bias term becomes:

$$b_2 = (1/1_M^T V_2 1_M)(-1_M^T \gamma_2 S_2 - 1_M^T V_2 \Omega \beta). \tag{5.44}$$

Substituting the obtained expression for the bias term b_2 into (5.43) along with some algebraic manipulation will result in the following linear system:

$$\gamma_5 \left(I_M - \frac{V_2 1_M 1_M^T}{1_M^T V_2 1_M} \right) S_2 = \beta - V_2 \left(I_M - \frac{1_M 1_M^T V_2}{1_M^T V_2 1_M} \right) \Omega \beta.$$

□

The proposed model selection criterion can be expressed as follows:

$$\underset{\gamma_1,\gamma_2,\gamma_3,\sigma}{\operatorname{argmax}} \kappa \, \mathrm{F}(\gamma_1, \gamma_2, \gamma_3, \sigma) + (1 - \kappa) \mathrm{Acc}(\gamma_1, \gamma_2, \gamma_3, \sigma)$$

which is an affine combination of Fisher criterion (F) [27] and classification accuracy (Acc). The Fisher criterion measures how localized the clusters appear. $\kappa \in [0, 1]$ is a user-defined parameter that controls the trade off between the importance given to unlabeled and labeled samples. In case few labeled samples are available one may give more weight to Fisher criterion and vice versa.

After completing the training stage and obtaining two non-parallel hyperplanes, the labels of the unseen test data points $\mathcal{X}_{test} = \{x_{test}^1, \ldots, x_{test}^n\}$ are determined depending on the perpendicular distances of the test points from the hyperplanes (see Fig. 5.2):

$$\widehat{\mathrm{Label}}(\mathcal{X}_{test}) = \arg\min_{k=1,2} \{d_k(\mathcal{X}_{test})\}, \tag{5.45}$$

where

$$d_k(\mathcal{X}_{test}) = \frac{|\Phi_{test} w_k + b_k 1_n|}{\|w_k\|_2}, k = 1, 2. \tag{5.46}$$

Here $\Phi_{test} = [\varphi(x_{test}^1), \ldots, \varphi(x_{test}^n)]^T$.

The procedure of the proposed non-parallel semi-supervised classification is outlined in Algorithm 7.

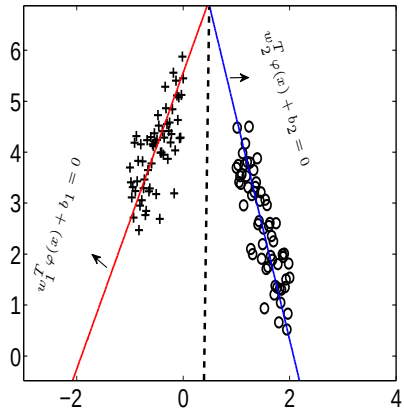


Figure 5.2: Data points of class I and II are marked by plus and circle signs respectively. The obtained non-parallel hyperplanes $w_1^T\varphi(x) + b_1 = 0$ and $w_2^T\varphi(x) + b_2 = 0$, are depicted by solid lines. The obtained decision boundary is shown by the dashed line.

Algorithm 7: Non-parallel semi-supervised KSC (NP-Semi-KSC)

Input: Training data set $\mathcal{X}$, labels $\mathcal{Y}$, the tuning parameters $\{\gamma_i\}_{i=1}^3$, the kernel bandwidth σ and the test set $\mathcal{X}_{test}$.

Output: Class membership of test data.

1: Solve the dual linear system (5.37) to obtain α and compute the bias term b_1 by (6.6). Therefore the first hyperplane (hypersurface) can be constructed.
2: Solve the dual linear system (6.3) to obtain β and compute the bias term b_2 by (5.44). Therefore the second hyperplane (hypersurface) can be constructed.
3: Compute the class membership of test data points using (5.45) and (5.46).

5.6 Numerical Experiments

5.6.1 Classification

In this section experimental results on a synthetic dataset called "cross-planes" and real-life datasets from the UCI machine learning repository [13] are given. We compare the performance of the proposed methods (LS-Hinge, LS-LS, LS-pinball) with classical LSSVMs and the method described in [8] over the above-mentioned datasets.

We first consider the cross-planes data set for the relationship between LSSVMs and LS-LS, which has been studied in Theorem 5.3.2. The obtained results are depicted in Figure 5.3. LSSVMs with linear kernel are first tuned on this data set to obtain the optimal regularization parameter γ. Then the obtained γ is fed into the LS-LS formulation as γ_1 and regularization parameter γ_2 is set to $\frac{\gamma_1}{r}$.

From Figure 5.3, it can be seen that the performance of the LS-LS when $r = 1$ ($\gamma_1 = \gamma_2$), is exactly equal to the performance of classical LSSVMs, i.e., we obtain two parallel hyperplanes. Whereas by changing the ratio r, which is defined as γ_1/γ_2, the classification accuracy is improved significantly. This is purely due to the ability of the proposed approach for designing two non-parallel hyperplanes. By changing the r value, hyperplanes start changing their directions. The optimal value for r is obtained by cross-validation method.

Figure 5.4, corresponds to the case when we have label noise, which can be regarded as outliers, in the data. As it was expected LS-LS is sensitive to noise whereas applying hinge or pinball loss functions will compensate the outliers to large extend.

For UCI data sets, the parameters, including regularization constants γ_1, γ_2, kernel bandwidth σ, and in the case of pinball loss the parameter τ, are obtained using Coupled Simulated Annealing [179] approach initialized with 5 random sets of parameters. On every iteration step for CSA method we proceed with a 10-fold cross-validation.

Descriptions of the used datasets from [13] can be found in Table 5.2. For Ecoli dataset some of the classes are merged in order to avoid unbalanced classes. One may consider the works reviewed in [153] to tackle the unbalanced classes.

We have artificially introduced random label and feature noise. To generate label noise, we randomly select 5% of samplings and change the observed labels. To generate feature noise, we add Gaussian noise to each feature and the signal-

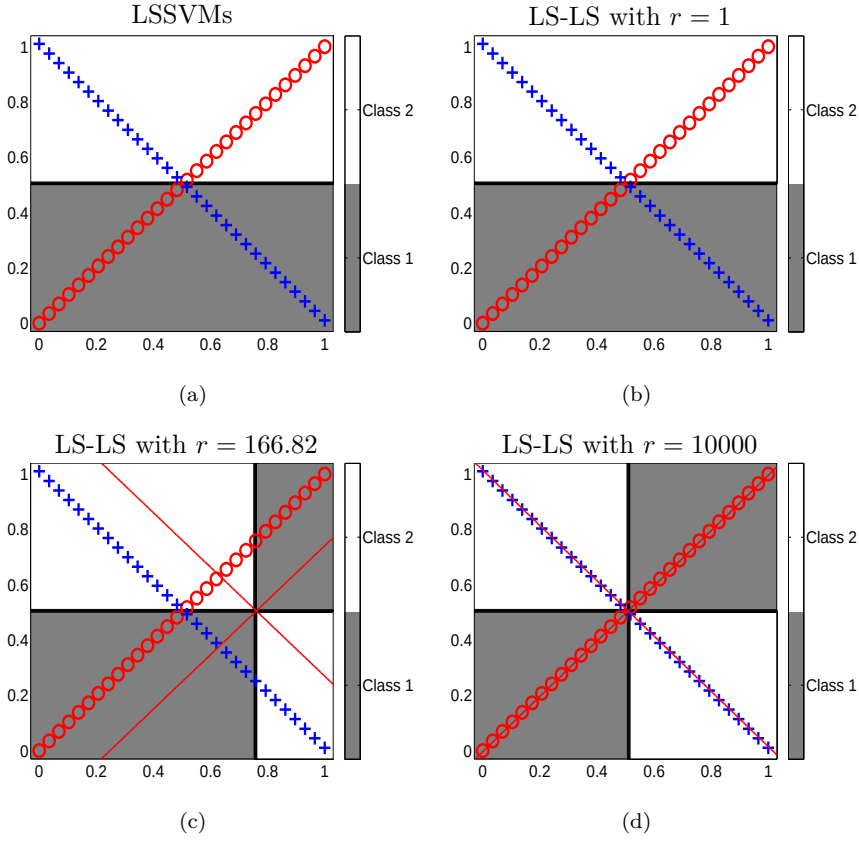


Figure 5.3: (a) Classification result obtained by LSSVMs with linear kernel, (b) Classification result obtained by LS-LS with linear kernel and $r = 1$, (c) Classification result obtained by LS-LS with linear kernel and $r = 166.82$, (d) Classification result obtained by LS-LS with linear kernel and $r = 10000$.

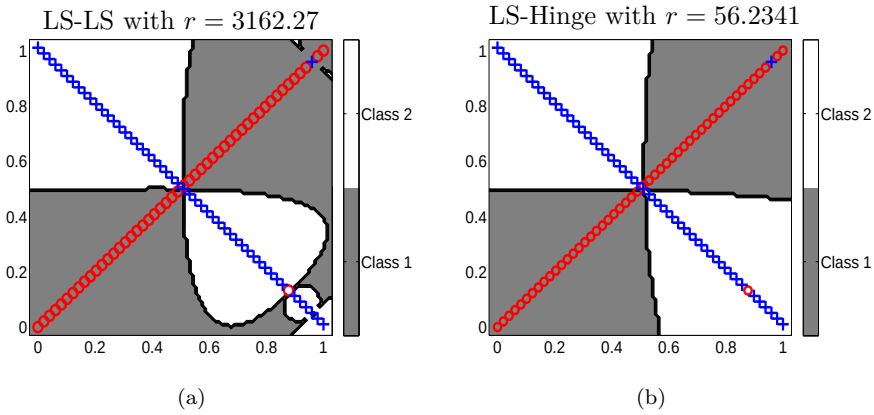


Figure 5.4: (a) Classification result obtained by LS-LS with nonlinear RBF kernel and, (b) Classification result obtained by LS-Hinge with nonlinear RBF kernel.

Table 5.2: Dataset statistics

Dataset	# training data	# testing data	# attributes	# classes
Iris	105	45	4	3
Spect	80	187	21	2
Heart	135	135	13	2
Ecoli	100	236	7	5
Monk1	124	432	6	2
Monk2	169	132	6	2
Monk3	122	432	6	2
Ionosphere	176	175	33	2
Spambase	500	4101	57	2
Magic	500	18520	10	2
Seeds	147	63	7	3
Wine	125	53	13	3

to-noise ratio is set to 20. All features for these data sets were normalized in a preprocessing step. We computed the means of the obtained accuracy over 10 simulation runs (every run includes 10 fold cross validation). The obtained results for RBF kernel are tabulated in Table 5.3, where the type of noise (no noise, label noise, feature noise, both label and feature noise), dimension of the data, and the size of the training and testing sets are reported.

As discussed previously, the proposed non-parallel SVMs have more flexibility than the classical SVMs. The advantage of non-parallel classifiers is more obvious in the linear kernel than in the RBF kernel case, since the RBF kernel itself provides enough flexibility for many cases. Therefore, in many

Table 5.3: Average binary classification accuracy on test sets with RBF kernel over 10 simulation runs with 5% label or/and feature noise

Datasets	Noise	**LSSVM** [157]	**LS-Hinge**	**LS-Pinball**	**LS-LS**	**LSTWSVM** [8]
Monk1	no noise	0.77	0.81	0.91	**0.96**	0.77
	label	0.78	**0.79**	**0.79**	0.78	0.78
	feature	0.72	**0.73**	0.72	0.72	0.64
	both	0.71	0.71	**0.73**	0.71	**0.73**
Monk2	no noise	0.87	0.86	0.87	**0.88**	**0.88**
	label	0.83	0.82	0.83	0.83	**0.84**
	feature	0.71	0.70	0.71	0.70	**0.72**
	both	0.69	**0.72**	**0.72**	0.70	0.71
Monk3	no noise	0.92	0.92	0.92	**0.93**	0.91
	label	0.90	0.91	**0.92**	0.90	0.88
	feature	0.85	0.85	**0.87**	0.83	0.81
	both	0.84	0.84	**0.86**	0.84	0.80
Spect	no noise	0.74	0.76	0.77	**0.84**	0.81
	label	0.77	**0.78**	0.75	0.77	0.77
	feature	0.71	0.77	0.74	0.78	**0.81**
	both	0.67	0.71	**0.77**	0.73	0.74
Ionosphere	no noise	**0.94**	**0.94**	**0.94**	**0.94**	0.93
	label	0.93	0.93	**0.94**	**0.94**	0.93
	feature	0.92	0.92	**0.93**	**0.93**	0.90
	both	0.89	0.92	**0.93**	**0.93**	0.92
Heart	no noise	0.83	0.82	0.81	**0.83**	0.70
	label	0.82	0.82	**0.82**	**0.82**	0.62
	feature	**0.86**	0.85	0.85	0.85	0.54
	both	0.82	0.82	0.82	**0.83**	0.63
Magic	no noise	0.78	**0.79**	**0.79**	0.78	0.59
	label	0.78	0.78	0.78	**0.79**	0.50
	feature	**0.78**	**0.78**	0.77	**0.78**	0.54
	both	0.77	0.71	0.77	**0.78**	0.51
Spambase	no noise	0.88	**0.91**	**0.91**	**0.91**	0.50
	label	0.89	**0.90**	**0.90**	**0.90**	0.50
	feature	0.88	0.88	**0.89**	**0.89**	0.51
	both	0.86	0.86	**0.88**	**0.88**	0.50

applications, the performance of classical SVMs and the non-parallel SVMs are similar. In Table 5.2, we only list the data sets with significant difference. The proposed non-parallel SVMs have different properties, due to the used loss functions. These properties have been discussed previously. The least squares error is insensitive to feature noise but could be significantly affected by large outliers. Hence LS-LS generally performs well in feature noise cases but not in label noise cases. The LSTWSVM is also a kind LS-LS scheme and has similar performance as LS-LS. In contrast, the hinge loss is robust to outliers but only a few samples contribute the classifier. In this way the obtained classifier is robust to label noise but is sensitive to feature noise. The property of pinball

loss used in classification has been discussed in [76]. Accordingly, LS-Pinball is a trade of between LS-LS and LS-Hinge and can give a good classifier when the data are contaminated by both label and feature noise.

The non-parallel framework (5.1) is proposed for multi-class problems. In the next experiment, we consider four data sets from UCI machine learning repository. As for binary case, four scenarios: no noise, label noise, feature noise and feature/label noise are investigated. The average classification accuracy on test sets over 10 simulation runs are tabulated in Table 5.4. The performance of the proposed schemes on multi-class problem coincides with our explanation for binary classification tasks.

Table 5.4: Average multi-class classification accuracy on test sets with RBF kernel over 10 simulation runs with 5% label or/and feature noise.

Datasets	Noise	**LSSVM** [157]	**LS-Hinge**	**LS-Pinball**	**LS-LS**	**LSTWSVM** [8]
Ecoli	no noise	**0.85**	0.84	0.84	**0.85**	0.84
	label	0.81	**0.82**	0.81	0.79	0.81
	feature	**0.83**	0.82	**0.83**	0.81	0.80
	both	0.78	0.77	**0.79**	**0.79**	0.76
Iris	no noise	**0.97**	0.96	0.96	0.94	0.96
	label	0.93	0.94	**0.95**	0.93	0.93
	feature	0.93	0.93	**0.94**	**0.94**	0.93
	both	0.93	0.92	**0.94**	0.93	0.89
Seeds	no noise	**0.95**	0.93	0.94	**0.95**	**0.95**
	label	0.93	**0.94**	**0.94**	0.91	0.92
	feature	0.92	0.92	0.93	**0.94**	0.92
	both	0.89	0.90	**0.91**	0.90	0.89
Wine	no noise	0.98	**0.99**	**0.99**	0.97	0.98
	label	0.97	0.98	**0.99**	0.96	0.98
	feature	0.97	**0.98**	**0.98**	**0.9**8	0.97
	both	0.97	**0.98**	**0.98**	0.97	0.97

5.6.2 Semi-supervised classification

The synthetic problem consist of four Gaussians with some overlap. The full dataset includes 200 data points. Each one of the training and validation sets consist of 100 points randomly selected form the entire dataset. Artificially binary labels are introduced for eight points.

The performance of the Semi-KSC [5] and the proposed method in this chapter when a linear kernel is used are shown in Fig. 5.5. Due to the ability of the method to produce two non-parallel hyperplanes the data points are almost correctly classified whereas Semi-KSC is not able to do the task well in this

case. This example can motivate the use of non-parallel Semi-KSC over Semi-KSC.

The performance of the method is also tested on some of the benchmark datasets for semi-supervised learning described in [41]. The benchmark consists of four data sets as shown in Table 5.5. The first two i.e. `g241c` and `g241d`, which consist of 1500 data points with dimensionality of 241, were artificially created. The other two datasets `BCI` and `Text` were derived from real data. All datasets have binary labels. `BCI` has 400 data points and dimension 117. `Text` includes 1500 data points with dimensionality 11960.

For each data set, 12 splits into labeled points and remaining unlabeled points is already provided (each split contains at least one point of each class). The tabulated results indicate the variability with respect to these 12 splits. In order to have a fair comparison with the recorded results in [5], for each split a training set of $N_{tr} = 150$ unlabeled samples are randomly selected for `BCI` data set and for all the other data sets $N_{tr} = 600$. The same number of unlabeled samples used in training sets are drawn at random to form the unlabeled samples of the validation sets. Among the labeled data points, 70% is used for training and 30% for the validation sets.

The result of the proposed method (NP-Semi-KSC) is compared with that of Semi-KSC, Laplacian SVM (LapSVM) [21] and its recent version LapSVMp [119] recorded in [5] over the datasets mentioned. When few labeled data points are available the proposed method shows a comparable result with respect to other methods. But as the number of labeled data points increases NP-Semi-KSC outperforms in most cases the other methods.

Table 5.5: Average misclassification test error $\times 100\%$. The calculation of the test error is done by evaluating the methods on the full data sets. Two cases for the labeled data size are considered (i.e. # labeled data points=10 and 100). In the case of 10 labeled data points, the performance of the proposed method is comparable to that of the other methods. When 100 labeled data points are used, the proposed method shows a better perfromance compared to LapSVM [21], LapSVMp [119] and Semi-KSC [5].

# of Labeled data	**Method**	`g241c`	`g241d`	`BCI`	`Text`
10	LapSVM	0.48 ± 0.02	0.42 ± 0.03	0.48 ± 0.03	0.37 ± 0.04
	LapSVMp	0.49 ± 0.01	0.43 ± 0.03	0.48 ± 0.02	0.40 ± 0.05
	Semi-KSC	$\mathbf{0.42 \pm 0.03}$	0.43 ± 0.04	$\mathbf{0.46 \pm 0.03}$	$\mathbf{0.29 \pm 0.06}$
	NP-Semi-KSC	0.44 ± 0.03	$\mathbf{0.41 \pm 0.02}$	0.47 ± 0.03	0.40 ± 0.05
100	LapSVM	0.40 ± 0.06	0.31 ± 0.03	0.37 ± 0.04	0.27 ± 0.02
	LapSVMp	0.36 ± 0.07	0.31 ± 0.02	0.32 ± 0.02	0.32 ± 0.02
	Semi-KSC	0.29 ± 0.05	0.28 ± 0.05	0.28 ± 0.02	$\mathbf{0.22 \pm 0.02}$
	NP-Semi-KSC	$\mathbf{0.23 \pm 0.01}$	$\mathbf{0.26 \pm 0.02}$	$\mathbf{0.26 \pm 0.01}$	0.23 ± 0.02

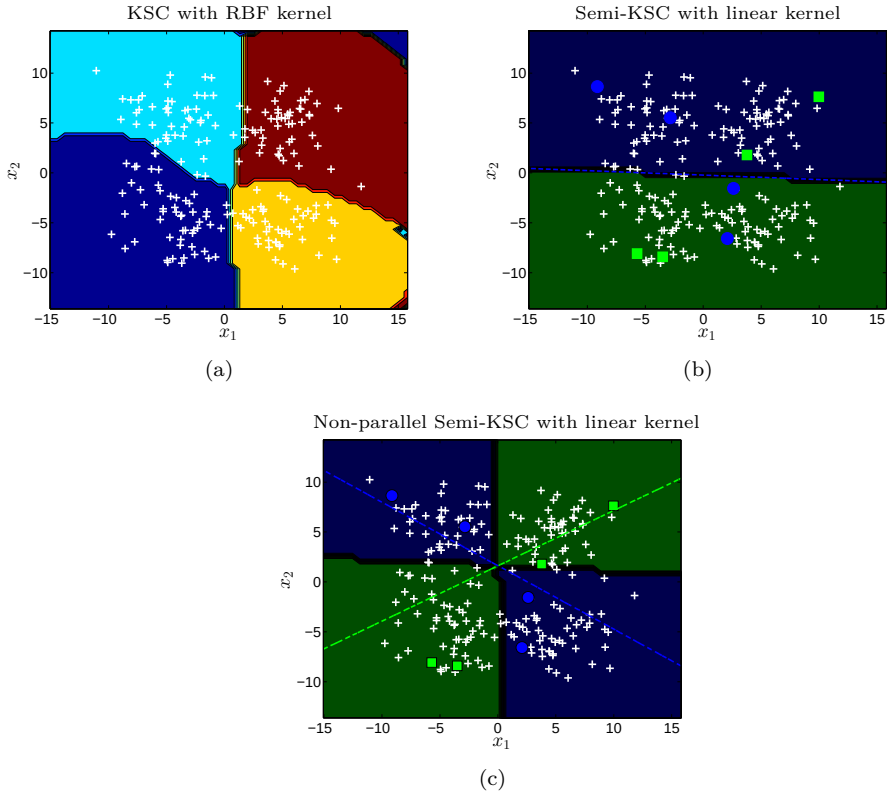


Figure 5.5: Toy Problem-Four Gaussians with some overlap. The training and validation parts consist of $N_{tr} = 100$ and $N_{val} = 100$ unlabeled data points respectively. The labeled data points of two classes are depicted by the blue squares and green circles. (a) Result of kernel spectral clustering (completely unsupervised). (b): Result of semi-supervised kernel spectral clustering when linear kernel is used. The separating hyperplane is shown by blue dashed line. (c): Result of the proposed non-parallel semi-supervised KSC when linear kernel is used. Two non- parallel hyperplanes are depicted by blue and green dashed lines.

5.7 Conclusions

In this chapter, a general framework for non-parallel classifier is proposed. As opposed to conventional approaches, the burden of formulating different optimization problems in the case of applying a non linear kernel, is avoided via utilizing the kernel trick in the dual. This framework enables the possibility of using different types of loss function. Generally, different loss functions perform well for different problems, which is supported by the numerical experiments. With the proposed non-parallel classifiers, one can choose the suitable loss functions and achieve satisfactory performance for different noise levels. Moreover, a non-parallel semi-supervised formulation based on kernel spectral clustering is developed. Semi-KSC formulation [5] is a special case of the proposed method when parameters of the new model are chosen appropriately. The validity and applicability of the proposed method is shown on synthetic examples as well as on real benchmark datasets.

Chapter 6

Semi-Supervised Learning

In this chapter, a multi-class semi-supervised learning algorithm using kernel spectral clustering (KSC) as a core model is proposed. A regularized KSC is formulated to estimate the class memberships of data points in a semi-supervised setting using the one-vs-all strategy while both labeled and unlabeled data points are present in the learning process. The propagation of the labels to a large amount of unlabeled data points is achieved by adding regularization terms to the cost function of the KSC formulation. In other words, imposing the regularization term enforces certain desired memberships. The model is then obtained by solving a linear system in the dual. Furthermore, the optimal embedding dimension is designed for semi-supervised clustering. This plays a key role when one deals with a large number of clusters. In addition, two approaches are proposed in order to make the algorithm scalable to large scale data sets where a huge amount of unlabeled data points is available. The highlights of this chapter can be summarized as follows:

- *Using an unsupervised model as the core model and incorporating the available side-information (labels) through a regularization term.*
- *Addressing both multi-class semi-supervised classification and semi-supervised clustering.*
- *Extension of the binary case to multi-class case and addressing the encoding schemes.*
- *Realizing low embedding dimension to reveal the existing number of clusters.*

- *Addressing scalability of the algorithm for dealing with large scale data sets.*

6.1 Related Work

The incorporation of some form of prior knowledge of the problem at hand into the learning process is a key element that allows an increase of performance in many applications.

In many contexts, ranging from data mining to machine perception, obtaining the labels of input data is often difficult and expensive. Therefore in many cases one deals with a huge amount of unlabeled data, while the fraction of labeled data points will typically be small.

Semi-supervised algorithms aim at learning from both labeled and unlabeled data points. In fact in semi-supervised learning one tries to incorporate the labels (prior knowledge) in the learning process to enhance the clustering/classification performance. Semi-supervised learning can be classified into two categories, i.e. transductive and inductive learning. Transductive learning aims at predicting the labels for a specified set of test data by taking both labeled and unlabeled data together into account in the learning process. In contrast, in inductive learning the goal is to learn a decision function from a training set consisting of labeled and unlabeled data for future unseen test data points. Throughout this chapter we refer to semi-supervised inductive learning as semi-supervised learning.

The semi-supervised inductive learning itself can be categorized into semi-supervised clustering and classification. The former addresses the problem of exploiting additional labeled data to adjust the cluster memberships of the unlabeled data. The latter aims at utilizing both unlabeled and labeled data to obtain a better classification model, and higher quality predictions on unseen test data points.

In some classical semi-supervised techniques, a classifier is first trained using the available labeled data points and then the labels for the unlabeled data points are predicted using out-of-extension. In the second step, unlabeled data that are classified with the highest confidence score are added incrementally to the training set and the process is repeated until the convergence is satisfactory [41, 186, 2]. Several semi-supervised algorithms have been proposed in the literature, see [77, 128, 21, 180, 181, 82, 174]. For instance, the Laplacian support vector machine (LapSVM) [21], is one of the graph based methods with a data-dependent geometric regularization which provides a natural out-

of-sample extension. The authors in [180] used local spline regression for semi-supervised classification by introducing splines developed in Sobolev space to map the data points to class labels. A transductive semi-supervised algorithm called ranking with Local Regression and Global Alignment (LRGA) to learn a robust Laplacian matrix for data ranking is proposed in [181]. In this approach, for each data point, the ranking scores of neighboring points are estimated using a local linear regression model. A label propagation approach in graph-based semi-supervised learning has been introduced in [82]. The authors in [174] developed a semi-supervised classification method based on class membership, motivated by the fact that similar instances should share similar label memberships.

Many semi-supervised algorithms perform well on relatively small problems, (see [41] and references therein), but they do not scale well to large datasets. Therefore turning semi-supervised learning algorithms into practice is important. For instance a family of semi-supervised linear support vector classifiers for large data sets is introduced in [148].

Spectral clustering methods belong to a family of unsupervised learning algorithms that make use of the eigenspectrum of the Laplacian matrix of the data to divide a dataset into natural groups such that points within the same group are similar and points in different groups are dissimilar to each other [127, 171, 48].

Kernel spectral clustering (KSC) is an unsupervised algorithm that represents a spectral clustering formulation as a weighted kernel PCA problem, cast in the LSSVM framework [4]. In contrast to classical spectral clustering, there is a systematic model selection scheme for tuning the parameters and also the extension of the clustering model to out-of-sample points is possible.

In [154], for the sake of dimensionality reduction, kernel maps with a reference point are generated from a least squares support vector machine core model via an additional regularization term for preserving local mutual distances together with reference point constraints. In contrast with the class of kernel eigenmap methods, the solution (coordinates in the low dimensional space) is characterized by a linear system instead of an eigenvalue problem.

Recently the authors in [5] have extended the kernel spectral clustering to binary semi-supervised learning (semi-KSC) by incorporating the information of labeled data points in the learning process. Therefore the problem formulation is a combination of unsupervised and binary classification approaches. Contrary to the approach described in [5], a non-parallel semi-supervised classification (NP-Semi-KSC) is introduced in [115]. It generates two non-parallel hyperplanes which are then used for out-of-sample extension.

It is the purpose of this chapter to develop a new Multi-class Semi-Supervised KSC-based algorithm (MSS-KSC) using a one-versus-all strategy. In contrast to the methods described in [41, 186, 2, 21, 180, 181, 82], in the proposed approach we start with a purely unsupervised algorithm as a core model and the available side information is incorporated via a regularization term. Given Q labels, the approach is not restricted to find just Q classes (semi-supervised classification) and instead it is able to uncover up to 2^Q hidden clusters (semi-supervised clustering). In addition, it uses low embedding dimension to reveal the existing number of clusters which is important when one deals with large number of clusters. There is a systematic model selection scheme for tuning the parameters and it is provided with the out-of-sample extension property. Furthermore the formulation is constructed for the multi-class semi-supervised classification and clustering.

Here KSC [4] is used as the core model. In this case thanks to the discriminative property of KSC one can benefit from unlabeled data points. Unlike the KSC approach that projects the data to a $k-1$ dimensional space for being able to group the data into k clusters, here the embedding dimension is rather equal to the number of available class-labels in the semi-supervised learning framework.

6.2 Semi-Supervised Classification

In this section we assume that there is a total number of Q classes ($\mathcal{C}_j$, $j = 1, \ldots, Q$). The corresponding number of available class labels is also equal to Q. Suppose the training data set $\mathcal{D}$ consists of M data points and is defined as follows

$$\mathcal{D} = \{ \underbrace{x_1, ..., x_N}_{\substack{Unlabeled\,data \\ (\mathcal{D}_U)}} , \underbrace{x_{N+1}, .., x_M}_{\substack{Labeled\,data \\ (\mathcal{D}_L)}} \}$$

where $\{x_i\}_{i=1}^M \in \mathbb{R}^d$. The labels are available for the last $N_L = M - N$ data points in $\mathcal{D}_L$ and are denoted by

$$Z = \left[z_{N+1}^T, \ldots, z_M^T\right]^T \in \mathbb{R}^{(M-N)\times Q},$$

with $z_i \in \{+1, -1\}^Q$ is the encoding vector for the training point x_i.

In the proposed method we start with an unsupervised algorithm as a core model. Then by introducing a regularization term, we incorporate the available side information, which in this case are the labels, to the core model. Here the kernel spectral clustering is used as the core model. Because as it has been shown in [4] in contrast to classical spectral clustering, KSC has a systematic

model selection scheme for tuning the parameters and it is provided with the out-of-sample extension property.

The one-vs-all strategy is utilized to build the codebook, i.e., the training points belonging to the i-th class are labeled by $+1$ and all the remaining data from the rest of the classes are considered to have negative labels. Both the labeled and unlabeled data points are arranged such that the top N data points are the unlabeled ones and the rest, i.e. N_L, are the labeled data points. We consider the labels of the unlabeled data points to be zero as in [5]. In our formulation unlabeled data points are only regularized using the KSC core model.

6.2.1 Primal-Dual formulation of the method

We formulate the multi-class semi-supervised learning in the primal as the following optimization problem [109]:

$$\begin{aligned} \min_{w^{(\ell)},b^{(\ell)},e^{(\ell)}} \quad & \frac{1}{2}\sum_{\ell=1}^{Q} w^{(\ell)^T} w^{(\ell)} - \frac{\gamma_1}{2}\sum_{\ell=1}^{Q} e^{(\ell)^T} V e^{(\ell)} + \\ & \frac{\gamma_2}{2}\sum_{\ell=1}^{Q} (e^{(\ell)} - c^{(\ell)})^T A (e^{(\ell)} - c^{(\ell)}) \\ \text{subject to} \quad & e^{(\ell)} = \Phi w^{(\ell)} + b^{(\ell)} 1_M,\ \ell = 1, \ldots, Q, \end{aligned} \tag{6.1}$$

where $c^{(\ell)}$ is the ℓ-th column of the matrix C defined as

$$C = [c^{(1)}, \ldots, c^{(Q)}]_{M\times Q} = \left[\frac{0_{N\times Q}}{Z}\right]_{M\times Q}, \tag{6.2}$$

where $0_{N\times Q}$ is a zero matrix of size $N \times Q$ and Z is defined as previously. $b^{(\ell)}$ is a bias term which is a scalar. The matrix A is defined as follows:

$$A = \left[\begin{array}{c|c} 0_{N\times N} & 0_{N\times N_L} \\ \hline 0_{N_L\times N} & I_{N_L\times N_L} \end{array}\right],$$

where $I_{N_L\times N_L}$ is the identity matrix of size $N_L \times N_L$.

The available prior knowledge, i.e. the labels, is added to the KSC model through the third term in the objective function of (6.1). This term aims at minimizing the difference between the score variables of the labeled data points, i.e. e_i for $i \in \mathcal{D}_L$, and the actual labels provided by the user. Therefore it enforces the e_i values for the labeled data points to be close enough to the actual labels in the projection space. Furthermore, since we do not intend to

prejudge about the memberships of the unlabeled data points, the matrix A is taking place in the third term in the objective function.

Lemma 6.2.1. *Given a positive definite kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $K(x, z) = \varphi(x)^T \varphi(z)$ and regularization constants $\gamma_1, \gamma_2 \in \mathbb{R}^+$, the solution to (6.1) is obtained by solving the following dual problem [109]:*

$$(I_M - RS\Omega)\alpha^{(\ell)} = \gamma_2 S^T c^{(\ell)}, \ell = 1, \ldots, Q, \tag{6.3}$$

where $R = \gamma_1 V - \gamma_2 A$, $\alpha^{(\ell)} = [\alpha_1^{(\ell)}, \ldots, \alpha_M^{(\ell)}]^T$ are the Lagrange multipliers and $S = I_M - (1/1_M^T R 1_M) 1_M 1_M^T R$. Ω and I_M are defined as previously.

Proof. The Lagrangian of the constrained optimization problem (6.1) becomes

$$\mathcal{L}(w^{(\ell)}, b^{(\ell)}, e^{(\ell)}, \alpha^{(\ell)}) = \frac{1}{2} \sum_{\ell=1}^{Q} {w^{(\ell)}}^T w^{(\ell)} - \frac{\gamma_1}{2} \sum_{\ell=1}^{Q} {e^{(\ell)}}^T V e^{(\ell)}$$

$$+ \frac{\gamma_2}{2} \sum_{\ell=1}^{Q} (e^{(\ell)} - c^{(\ell)})^T A (e^{(\ell)} - c^{(\ell)}) +$$

$$\sum_{\ell=1}^{Q} {\alpha^{(\ell)}}^T \left(e^{(\ell)} - \Phi w^{(\ell)} - b^{(\ell)} 1_M \right),$$

where $\alpha^{(\ell)}$ is the vector of Lagrange multipliers. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w^{(\ell)}} = 0 \to w^{(\ell)} = \Phi^T \alpha^{(\ell)}, \ell = 1, \ldots, Q, \\ \frac{\partial \mathcal{L}}{\partial b^{(\ell)}} = 0 \to 1_M^T \alpha^{(\ell)} = 0, \ell = 1, \ldots, Q, \\ \frac{\partial \mathcal{L}}{\partial e^{(\ell)}} = 0 \to \alpha^{(\ell)} = (\gamma_1 V - \gamma_2 A) e^{(\ell)} + \gamma_2 c^{(\ell)}, \ell = 1, \ldots, Q, \\ \frac{\partial \mathcal{L}}{\partial \alpha^{(\ell)}} = 0 \to e^{(\ell)} = \Phi w^{(\ell)} + b^{(\ell)} 1_M, \ell = 1, \ldots, Q. \end{cases} \tag{6.4}$$

Elimination of the primal variables $w^{(\ell)}, e^{(\ell)}$ and making use of Mercer's Theorem [167], results in the following equation

$$R\Omega\alpha^{(\ell)} + b^{(\ell)} R 1_M = \alpha^{(\ell)} - \gamma_2 c^{(\ell)}, \ell = 1, \ldots, Q, \tag{6.5}$$

where $R = \gamma_1 V - \gamma_2 A$. From the second KKT optimality condition and (6.5), the bias term becomes:

$$b^{(\ell)} = (1/1_M^T R 1_M)(-1_M^T \gamma_2 c^{(\ell)} - 1_M^T R \Omega \alpha^{(\ell)}), \ell = 1, \ldots, Q. \tag{6.6}$$

Substituting the obtained expression for the bias term $b^{(\ell)}$ into (6.5) along with some algebraic manipulation one can obtain the solution in dual as the following linear system:

$$\gamma_2\left(I_M - \frac{R1_M1_M^T}{1_M^TR1_M}\right)c^{(\ell)} = \alpha^{(\ell)} - R\left(I_M - \frac{1_M1_M^TR}{1_M^TR1_M}\right)\Omega\alpha^{(\ell)}.$$

□

Remark 6.2.1. *It should be noted that since the optimization problem (6.1) does have equality constraints therefore the KKT conditions include the primal equality constraints and the gradient of the Lagrangian with respect to the primal variables (see [32, Chapter 5]). In (6.4), the first three equations correspond to the derivative of the Lagrangian with respect to primal variables and the primal equality constraints are equivalently obtained by taking the derivative of the Lagrangian with respect to dual variables.*

It should be noticed that one can also obtain the following linear system when the primal variables $w^{(\ell)}, e^{(\ell)}$ are eliminated from (KKT) optimality conditions in (6.4):

$$\left[\begin{array}{c|c} \Omega - R^{-1} & 1_M \\ \hline 1_M^T & 0 \end{array}\right]\left[\begin{array}{c} \alpha^{(\ell)} \\ \hline b^{(\ell)} \end{array}\right] = \left[\begin{array}{c} -R^{-1}\gamma_2c^{(\ell)} \\ \hline 0 \end{array}\right], \ \ell = 1, \ldots, Q, \tag{6.7}$$

where $\alpha^{(\ell)} = [\alpha_1^{(\ell)}, \ldots, \alpha_M^{(\ell)}]^T$ and $\Omega = \Phi\Phi^T$ is the kernel matrix. Matrix R is a diagonal matrix and it is invertible if and only if $\gamma_1 v_i \neq \gamma_2$ for $i = 1, \ldots, M$.

The linear systems (6.3) and (6.7) have a unique solution when the associated coefficient matrix is full-rank which depends on the regularization parameters.

6.2.2 Encoding/Decoding scheme

In semi-supervised classification, the encoding scheme is chosen in advance since the number of existing classes is known beforehand. The codebook $\mathcal{CB}$ used for out-of-sample extension is defined based on the encoding vectors for the training points. If $Z = [z_{N+1}^T, \ldots, z_M^T]^T$ is the encoding matrix for the training points, the $\mathcal{CB} = \{c_q\}_{q=1}^Q$, where $c_q \in \{-1, 1\}^Q$, is defined by the unique rows of Z (i.e. from identical rows of Z one selects one row). Considering the test

set $\mathcal{D}^{test} = \{x_i^{test}\}_{i=1}^{N_{test}}$ the score variables evaluated at the test points become:

$$\begin{aligned} e_{test}^{(\ell)} &= \Phi_{test} w^{\ell} + b^{(\ell)} 1_{N_{test}} \\ &= \Omega_{test} \alpha^{(\ell)} + b^{(\ell)} 1_{N_{test}},\ \ell = 1, \ldots, Q, \end{aligned} \tag{6.8}$$

where $\Omega_{test} = \Phi_{test} \Phi^T$. The procedure for the multi-class semi-supervised classification is summarized in Algorithm 8.

Algorithm 8: Multi-class semi-supervised classification

Input: Training data set $\mathcal{D}$, labels Z, tuning parameters $\{\gamma_i\}_{i=1}^{2}$, kernel parameter (if any), test set $\mathcal{D}^{test} = \{x_i^{test}\}_{i=1}^{N_{test}}$ and codebook $\mathcal{CB} = \{c_q\}_{q=1}^{Q}$

Output: Class membership of test data points $\mathcal{D}^{test}$

1 Solve the dual linear system (6.3) to obtain $\{\alpha^{\ell}\}_{\ell=1}^{Q}$ and compute the bias term $\{b^{\ell}\}_{\ell=1}^{Q}$ using (6.6).

2 Estimate the test data projections $\{e_{test}^{(\ell)}\}_{\ell=1}^{Q}$ using (6.8).

3 Binarize the test projections and form the encoding matrix $[\text{sign}(e_{test}^{(1)}), \ldots, \text{sign}(e_{test}^{(Q)})]_{N_{test} \times Q}$ for the test points (Here $e_{test}^{(\ell)} = [e_{test,1}^{(\ell)}, \ldots, e_{test,N_{test}}^{(\ell)}]^T$).

4 $\forall i$, assign x_i^{test} to class q^*, where $q^* = \underset{q}{\text{argmin}}\, d_H(e_{test,i}^{\ell}, c_q)$ and $d_H(\cdot,\cdot)$ is the Hamming distance.

6.3 Semi-Supervised Clustering

In what follows we assume that there is a total number of T clusters and a few labels from Q of the clusters are available ($Q \leq T$). Therefore we are dealing with the case that some of the clusters are partially labeled. The aim is to incorporate these labels in the learning process to guide the clustering algorithm to adjust the membership of the unlabeled data. Next we will show how one can use the approach described in section 6.2 in this setting.

6.3.1 From solution of linear systems to clusters: encoding

Since the number of existing clusters is not known a priori, one cannot use the predefined codebook as in semi-supervised classification. Therefore a new

scheme is developed for generating a codebook to be used in the learning process.

It has been observed that the solution vector $\alpha^{(\ell)}$, $\ell = 1, \ldots, Q$ of the dual linear system (6.3) has a piecewise constant property when there is an underlying cluster structure in the data (see Fig. 2(d)). Once the solution to (6.3) is found, the codebook $\mathcal{CB} \in \{-1, 1\}^{p \times Q}$ is formed by the unique rows of the binarized solution matrix (i.e. $[\text{sign}(\alpha^{(1)}), \ldots, \text{sign}(\alpha^{(Q)})]$). The maximum number of clusters that can be decoded is 2^Q since the maximum value that p can take is 2^Q. In our approach the number of encodings, i.e. p, is tuned along with the model selection procedure. Therefore a grid search on the interval $[Q, 2^Q]$ is conducted to determine the number of clusters.

It should be noted that in Algorithm 8, the static codebook $\mathcal{CB}$ (static in a sense that the number of codewords is fixed and only depends on Q) is known beforehand and is of size $Q \times Q$. On the other hand in Algorithm 9, the codebook $\mathcal{CB}$ is no longer a static codebook and is of size $p \times Q$, where p can be maximally 2^Q. Furthermore, it is obtained based on the solution matrix S_α (see steps 2 and 3 in Algorithm 9).

6.3.2 Low dimensional spectral embedding

One may notice that as opposed to kernel spectral clustering [4] where the score variables lie in a $T - 1$ (where T is the actual number of clusters) dimensional space, in our formulation the embedding dimension is Q which can be smaller than T. This can also be seen as the optimized embedding dimension for clustering which plays an important role when the number of existing clusters is large. In fact one only requires $Q = \log T$ solution vectors to uncover T clusters. Therefore one is able to deal with a larger number of clusters in a more compact way. In contrast with the KSC approach where one needs to solve an eigenvalue problem, in MSS-KSC formulation one solves a linear system. It should be noted that although the two approaches share almost the same computational complexity, the quality of the solution vector obtained by the proposed algorithm is higher than that of KSC as shown in Fig. 6.6 and 6.7. This demonstrates the advantage of prior knowledge incorporation. The proposed semi-supervised clustering is summarized in Algorithm 9.

Algorithm 9: Semi-supervised clustering

Input: Training data set $\mathcal{D}$, labels Z, the tuning parameters $\{\gamma_i\}_{i=1}^{2}$, the kernel parameter (if any), number of clusters k, the test set $\mathcal{D}^{test} = \{x_i^{test}\}_{i=1}^{N_{test}}$ and number of available class labels i.e. Q

Output: Cluster membership of test data points $\mathcal{D}^{test}$

1 Solve the dual linear system (6.3) to obtain $\{\alpha^{\ell}\}_{\ell=1}^{Q}$ and compute the bias term $\{b^{\ell}\}_{\ell=1}^{Q}$ using (6.6).

2 Binarize the solution matrix $S_\alpha = [\text{sign}(\alpha^{(1)}), \ldots, \text{sign}(\alpha^{(Q)})]_{M \times Q}$, where $\alpha^{\ell} = [\alpha_1^{\ell}, \ldots, \alpha_M^{\ell}]^T$.

3 Form the codebook $\mathcal{CB} = \{c_q\}_{q=1}^{p}$, where $c_q \in \{-1, 1\}^Q$, using the k most frequently occurring encodings from unique rows of solution matrix S_α.

4 Estimate the test data projections $\{e_{test}^{(\ell)}\}_{\ell=1}^{Q}$ using (6.8).

5 Binarize the test projections and form the encoding matrix $[\text{sign}(e_{test}^{(1)}), \ldots, \text{sign}(e_{test}^{(Q)})]_{N_{test} \times Q}$ for the test points (Here $e_{test}^{\ell} = [e_{test,1}^{\ell}, \ldots, e_{test,N_{test}}^{\ell}]^T$).

6 $\forall i$, assign x_i^{test} to class/cluster q^*, where $q^* = \underset{q}{\text{argmin}}\, d_H(e_{test,i}^{\ell}, c_q)$ and $d_H(\cdot, \cdot)$ is the Hamming distance.

6.4 Model Selection

The performance of the multi-class semi-supervised model depends on the choice of the tuning parameters. In the case of RBF kernel the optimal values of γ_1, γ_2 and the kernel parameter σ can be obtained by evaluating the performance of the model (classification accuracy) on the validation set using a grid search over the parameters. One may also consider to utilize Coupled Simulated Annealing (CSA) in order to minimize the misclassification error in the cross-validation process. CSA leads to an improved optimization efficiency due to the fact that it reduces the sensitivity of the algorithm with respect to the initialization of the parameters while guiding the optimization process to quasi-optimal runs [179].

In the experiments, based on the analysis given in [5, Section III.C] we set $\gamma_1 = 1$. Then γ_2 and σ are tuned through a grid search. The range in which the search is made is discussed for each of the experiments in section VI. In general in the experiments we observed that a good value for γ_2, most of the times, is selected from the range $[0, 1]$.

Since labeled and unlabeled data points are involved in the learning process, it is natural to have a model selection criterion that makes use of both. Therefore,

for semi-supervised classification, one may combine two criteria where one of them evaluates the performance of the model on the unlabeled data points (evaluation of clustering results) and the other one maximizes the classification accuracy [5, 115].

A common approach for evaluating the quality of the clustering results consists of using internal cluster validity indices [24] such as Silhouette, Fisher and Davies-Bouldin index (DB) criteria. Here the Silhouette index is used to assess the clustering results. The Silhouette technique assigns to the ith sample of j-th class, $\mathcal{C}_j$, a quality measure $s(i)$ which is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$

Here $a(i)$ is the average distance between the i-th sample and all of the samples included in $\mathcal{C}_j$. b_i is the minimum average distance from the i-th sample to points in different clusters. The silhouette value for each sample is a measure of how similar that sample is to samples in its own cluster versus samples in other clusters, and is in the range of $[-1, 1]$.

The proposed model selection criterion for semi-supervised learning, with kernel parameter σ, can be expressed as follows:

$$\max_{\gamma_1,\gamma_2,\sigma,k} \eta\,\mathrm{Sil}(\gamma_1, \gamma_2, \sigma, k) + (1 - \eta)\mathrm{Acc}(\gamma_1, \gamma_2, \sigma, k). \tag{6.9}$$

It is a convex combination of Silhouette (Sil) and classification accuracy (Acc). $\eta \in [0, 1]$ is a user-defined parameter that controls the trade off between the importance given to unlabeled and labeled instances. In case few labeled data points are available one may give more weight to Silhouette criterion and vice versa.

The silhouette criterion is evaluated on the unlabeled data points in the validation set. One can also consider to evaluate it on the out-of-sample solution vectors.

In equation (6.9), k denotes the number of clusters that is unknown beforehand. In the case of semi-supervised classification where the number of classes is known a priori, one does not need to tune k and thus it can be removed from the list of decision variables of the aforementioned model selection criterion.

In any unsupervised learning algorithm one has to find the right number of existing clusters over the specified range which is provided by the user. When there is a form of prior knowledge about the data under study, the search space is reduced. In our semi-supervised clustering the lower bound of the range in which the number of clusters are sought is Q (assuming that Q cluster labels

are available). Therefore applying the proposed MSS-KSC algorithm will make it easier to reveal the lower level of the cluster hierarchy. In the proposed MSS-KSC approach one requires to solve a linear system. Therefore the complexity of the proposed MSS-KSC algorithm in the worst case scenario is $\mathcal{O}(M^3)$ where M is the number of training data points.

6.5 Large Scale Semi-Supervised Learning

Considering the large amount of unlabeled data, making a semi-supervised algorithm scalable is an important task. In this section I adopt the MSS-KSC algorithm and make it scalable. To this end, two possible schemes are proposed.

- The first approach, which will be referred to as Fixed-Size MSS-KSC (FS-MSS-KSC), is based on the Nyström approximation and the primal-dual formulation of the MSS-KSC. This is done by using a sparse approximation of the nonlinear mapping induced by the kernel matrix and solving the problem in the primal.
- The second approach is by means of the reduced kernel technique that solves the problem in the dual by reducing the dimensionality of the kernel matrix to a rectangular kernel. The second approach will be referred to as Reduced MSS-KSC (RD-MSS-KSC) approach.

6.5.1 Approximation to the feature map

In order to handle large data sets the so called fixed-size approach, where the feature map is approximated by the Nyström method [177, 16], is introduced in [156] and has been applied in [57, 51]. In what follows, the fixed-size approach is briefly summarized.

The approach is based on the fact that one can obtain an explicit expression finite dimension for the feature map $\varphi(\cdot)$ by means of an eigenvalue decomposition of the kernel matrix Ω. Consider the Fredholm integral equation of the first kind:

$$\int_C K(x, x_j)\phi_i(x)d\mu(x) = \lambda_i\phi_i(x_j) \tag{6.10}$$

where, for the sake of simplicity, C is a compact subset of $\mathbb{R}^d$, the kernel $K(.,.)$ is continuous on $C \times C$ and μ is a probability measure on C (see e.g. [62] for more details). Given a finite sample $\{x_j\}_{j=1}^M$ distributed according to μ, the approximation of the eigenfunction $\phi_i(x)$ in (6.10) can be obtained by the

Nyström method which applies a quadrature rule for discretizing the left-hand side of (6.10). This will lead to the eigenvalue problem [177]:

$$\frac{1}{M}\sum_{k=1}^{M} K(x_k, x_j)u_{ik} = \lambda_i^{(s)}\, u_{ij} \tag{6.11}$$

where here measure μ is approximated by $\frac{1}{M}\sum_i \delta_{x_i}$. The eigenvalues λ_i and eigenfunctions ϕ_i from the continuous problem (6.10) can be approximated by the sample eigenvalues $\lambda_i^{(s)}$ and eigenvectors u_i. Therefore, the i-th component of the feature map $\hat{\varphi} : \mathbb{R}^d \to \mathbb{R}^M$, for any point $x \in \mathbb{R}^d$, can be obtained as follows:

$$\hat{\varphi}_i(x) = \frac{1}{\lambda_i^{(s)}}\sum_{k=1}^{M} u_{ki}\, K(x_k, x) \tag{6.12}$$

where $\lambda_i^{(s)}$ and u_i are eigenvalues and eigenvectors of the kernel matrix $\Omega_{M\times M}$. Furthermore, the k-th element of the i-th eigenvector is denoted by u_{ki}. In practice when M is large, one works with a subsample (prototype vectors) of size $m \ll M$ whose elements are selected using an entropy based criterion. The entropy criterion ensures that the selected subset is spread over the entire data region and not only concentrated on a certain area of the data set. In this case, the m-dimensional feature map $\hat{\varphi} : \mathbb{R}^d \to \mathbb{R}^m$ can be approximated as follows:

$$\hat{\varphi}(x) = [\hat{\varphi}_1(x), \ldots, \hat{\varphi}_m(x)]^T \tag{6.13}$$

where

$$\hat{\varphi}_i(x) = \frac{1}{\lambda_i^{(s)}}\sum_{k=1}^{m} u_{ki}\, K(x_k, x), i = 1, \ldots, m \tag{6.14}$$

where $\lambda_i^{(s)}$ and u_i are now eigenvalues and eigenvectors of the constructed kernel matrix $\Omega_{m\times m}$ using the selected prototype vectors.

6.5.2 Fixed-Size MSS-KSC for large scale datasets

Since in Equation (6.1) the feature map φ is not explicitly known, one uses the kernel trick and solves the problem in the dual. But as it has been shown in subsection 6.2.1 in the dual one has to solve a linear system of size M (number of data points). Therefore for large scale data, it is not appropriate to solve the problem in the dual. In what follows we will show how one can use the approximation of the feature map to solve the problem in primal. Given the finite dimensional (m-dimensional) approximation to the feature map, i.e.

$$\hat{\Phi} = [\hat{\varphi}(x_1), \ldots, \hat{\varphi}(x_M)]^T \in \mathbb{R}^{M\times m} \tag{6.15}$$

one can rewrite the above optimization problem as an unconstrained optimization problem and solve it in primal:

$$\min_{w^{(\ell)},b^{(\ell)}} J(w^{(\ell)},b^{(\ell)}) = \frac{1}{2}\sum_{\ell=1}^{Q} {w^{(\ell)}}^T w^{(\ell)} -$$

$$\frac{\gamma_1}{2}\sum_{\ell=1}^{Q}(\hat{\Phi}w^{(\ell)} + b^{(\ell)}{1_M}^T)^T V(\hat{\Phi}w^{(\ell)} + b^{(\ell)}1_M) + \tag{6.16}$$

$$\frac{\gamma_2}{2}\sum_{\ell=1}^{Q}(c^{(\ell)} - \hat{\Phi}w^{(\ell)} + b^{(\ell)}1_M)^T A(c^{(\ell)} - \hat{\Phi}w^{(\ell)} + b^{(\ell)}1_M)$$

where the matrix C is defined as previously.

Lemma 6.5.1. *Given a finite dimensional (m-dimensional) approximation to the feature map $\hat{\Phi}$ and regularization constants $\gamma_1, \gamma_2 \in \mathbb{R}^+$, the solution to (6.16) is obtained by solving the following linear system [117]:*

$$\begin{bmatrix} w^{(\ell)} \\ b^{(\ell)} \end{bmatrix} = \left(\Phi_e^T R \Phi_e + I_{(m+1)}\right)^{-1} \gamma_2 \Phi_e^T c^{(\ell)}, \ell = 1, \ldots, Q, \tag{6.17}$$

where $R = \gamma_2 A - \gamma_1 V$ is a diagonal matrix, $\Phi_e^T = \begin{bmatrix} \hat{\Phi}^T \\ 1_M^T \end{bmatrix}_{(m+1)\times M}$ and $I_{(m+1)}$ is the identity matrix of size $(m+1)\times(m+1)$.

Proof. Taking the derivative of the cost function J with respect to $w^{(\ell)}$ and $b^{(\ell)}$ yields:

$$\begin{cases} \frac{\partial \mathcal{J}}{\partial w^{(\ell)}} = 0 \rightarrow \\ (I + \hat{\Phi}^T R \hat{\Phi}) w^{(\ell)} + \hat{\Phi}^T R 1_M b^{(\ell)} = \gamma_2 \hat{\Phi}^T c^{(\ell)}, \ \ell = 1, \ldots, Q, \\ \frac{\partial \mathcal{L}}{\partial b^{(\ell)}} = 0 \rightarrow \\ 1_M^T R \hat{\Phi} w^{(\ell)} + (1_M^T R 1_M) b^{(\ell)} = \gamma_2 1_M^T c^{(\ell)}, \ \ell = 1, \ldots, Q, \end{cases} \tag{6.18}$$

which then by using some algebraic manipulation can be rewritten as in (6.17). □

The codebook $\mathcal{CB}$ used for out-of-sample extension is defined based on the encoding vectors for the training points. If Y is the encoding matrix for the

training points, the $\mathcal{CB} = \{c_q\}_{q=1}^{Q}$, where $c_q \in \{-1, 1\}^Q$, is defined by the unique rows of Y (i.e. from identical rows of Y one selects one row). The score variables evaluated at the test set $\mathcal{D}^{\text{test}} = \{x_i\}_{i=1}^{n_{\text{test}}}$ become:

$$e_{\text{test}}^{(\ell)} = \hat{\Phi}_{\text{test}} w^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}} \; \ell = 1, \ldots, Q, \tag{6.19}$$

where $\hat{\Phi}_{\text{test}} = [\hat{\varphi}(x_1), \ldots, \hat{\varphi}(x_{n_{\text{test}}})]^T \in \mathbb{R}^{n_{\text{test}} \times m}$.

The decoding scheme consists of comparing the binarized score variables for test data points with the codebook $\mathcal{CB}$ and selecting the nearest codeword in terms of Hamming distance. The computational complexity, neglecting lower order terms, for solving linear system (6.17) is approximately $\mathcal{O}(m^3 + Mm^2)$ with $m \ll M$. (The complexity of calculating the Nyström approximation $\mathcal{O}(m^3 + m^2 M)$ is also included).

The procedure for the Fixed-Size MSS-KSC approach is summarized in Algorithm 10.

Algorithm 10: Fixed-size MSS-KSC approach for large scale data

Input: Training data set $\mathcal{D}$, labels Y, tuning parameters γ_1 and γ_2, kernel parameter (if any), test set $\mathcal{D}^{\text{test}} = \{x_i\}_{i=1}^{n_{\text{test}}}$ and codebook $\mathcal{CB} = \{c_q\}_{q=1}^{Q}$

Output: Class membership of test data points $\mathcal{D}^{\text{test}}$

1 Select m prototype vectors (small working set) using quadratic Rényi entropy criterion [64]. (see section IV. B)
2 Obtain the m-dimensional approximation of the feature map (6.15) by means of Nyström approximation (6.14).
3 Compute $\{w^{(\ell)}\}_{\ell=1}^{Q}$ and the bias term $\{b^{(\ell)}\}_{\ell=1}^{Q}$ using (6.17).
4 Estimate the test data projections $\{e_{\text{test}}^{(\ell)}\}_{\ell=1}^{Q}$ using (6.19).
5 Binarize the test projections and form the encoding matrix $[\text{sign}(e_{\text{test}}^{(1)}), \ldots, \text{sign}(e_{\text{test}}^{(Q)})]_{n_{\text{test}} \times Q}$ for the test points (Here $e_{\text{test}}^{(\ell)} = [e_{\text{test},1}^{(\ell)}, \ldots, e_{\text{test},n_{\text{test}}}^{(\ell)}]^T$).
6 $\forall i \, (i = 1, \ldots, n_{\text{test}})$, assign x_i to class q^*, where $q^* = \underset{q}{\text{argmin}} \, d_H(e_{\text{test},i}^{\ell}, c_q)$ and $d_H(\cdot, \cdot)$ is the Hamming distance.

6.5.3 Subsample selection for Nyström approximation

We aim at using an m-dimensional approximation to the feature map φ. Therefore as it is explained in subsection 6.5.1, one needs to select a subset

of fixed size m from a pool of training points of size M. Since the training set is composed of labeled and unlabeled data points, we select a subset (of size m) such that it consists of m_1 and m_2 data points from labeled and unlabeled training data points. ($m = m_1 + m_2$). As it has been motivated in [156], the Rényi entropy criterion [64] is used, twice only, to select m_1 points from the labeled and m_2 points from the unlabeled training data. Once the subset is available, the m-dimensional feature map is obtained using equation (6.14). See Fig. 6.11 as an application of FS-MSS-KSC approach on two moons problems.

6.5.4 Reduced MSS-KSC for large scale datasets

For large-scale problems, the difficulty of solving the MSS-KSC formulation (6.1) in the dual results from the huge kernel matrix which cannot be stored into memory. The authors in [94] proposed to restrict the number of support vectors by solving the reduced support vector machines (RSVM) for classification problem. The reduced kernel technique is utilized to reduce the $M \times M$ dimensionality of the kernel Ω to a much smaller $M \times \bar{n}$ dimensionality. Here $\bar{n}$ is the size of a randomly selected subset of training data considered as candidates of support vectors. A smaller matrix then can be stored into memory.

In what follows, the reduced kernel technique described in [94] is applied to the MSS-KSC formulation (6.1) in order to make it scalable. Suppose the matrix of training data points which includes both labeled and unlabeled samples is denoted by:

$$X = [x_1, \ldots, x_M]^T \in \mathbb{R}^{M \times d}.$$

Let us start with a linear kernel and reformulate (6.1) as follows:

$$\begin{aligned} \min_{w^{(\ell)}, b^{(\ell)}, e^{(\ell)}} \quad & \frac{1}{2} \sum_{\ell=1}^{Q} \left({w^{(\ell)}}^T w^{(\ell)} + (b^{(\ell)})^2 \right) - \frac{\gamma_1}{2} \sum_{\ell=1}^{Q} {e^{(\ell)}}^T V e^{(\ell)} \\ & + \frac{\gamma_2}{2} \sum_{\ell=1}^{Q} (e^{(\ell)} - c^{(\ell)})^T A (e^{(\ell)} - c^{(\ell)}) \\ \text{subject to} \quad & e^{(\ell)} = X w^{(\ell)} + b^{(\ell)} 1_M, \ \ell = 1, \ldots, Q, \end{aligned} \tag{6.20}$$

where here the bias term is also penalized just to make the subsequent derivations simpler. Setting the gradient of the associated Lagrangian of (6.20) with respect to $w^{(\ell)}$ to zero gives the following KKT condition:

$$w^{(\ell)} = X^T \alpha^{(\ell)}, \tag{6.21}$$

where $\alpha^{(\ell)}$ are the Lagrange multipliers associated with the equality constraint of (6.20). By replacing the primal variables $w^{(\ell)}$ from (6.21) one obtains:

$$\begin{aligned} \min_{\alpha^{(\ell)},b^{(\ell)},e^{(\ell)}} \quad & \frac{1}{2}\sum_{\ell=1}^{Q}\left({\alpha^{(\ell)}}^T\alpha^{(\ell)} + (b^{(\ell)})^2\right) - \frac{\gamma_1}{2}\sum_{\ell=1}^{Q}{e^{(\ell)}}^T V e^{(\ell)} \\ & + \frac{\gamma_2}{2}\sum_{\ell=1}^{Q}(e^{(\ell)} - c^{(\ell)})^T A(e^{(\ell)} - c^{(\ell)}) \\ \text{subject to} \quad & e^{(\ell)} = XX^T\alpha^{(\ell)} + b^{(\ell)}1_M,\ \ell = 1,\ldots,Q, \end{aligned} \tag{6.22}$$

where the objective function is modified to have the L2 norm regularization of the problem variables $\alpha^{(\ell)}, b^{(\ell)}, e^{(\ell)}$. Following the lines of [94] one can now replace the linear kernel matrix XX^T by a nonlinear kernel matrix with elements $\Omega_{ij} = K(x_i, x_j)$ to obtain the following optimization problem:

$$\begin{aligned} \min_{\alpha^{(\ell)},b^{(\ell)},e^{(\ell)}} \quad & \frac{1}{2}\sum_{\ell=1}^{Q}\left({\alpha^{(\ell)}}^T\alpha^{(\ell)} + (b^{(\ell)})^2\right) - \frac{\gamma_1}{2}\sum_{\ell=1}^{Q}{e^{(\ell)}}^T V e^{(\ell)} \\ & + \frac{\gamma_2}{2}\sum_{\ell=1}^{Q}(e^{(\ell)} - c^{(\ell)})^T A(e^{(\ell)} - c^{(\ell)}) \\ \text{subject to} \quad & e^{(\ell)} = \Omega\alpha^{(\ell)} + b^{(\ell)}1_M,\ \ell = 1,\ldots,Q. \end{aligned} \tag{6.23}$$

Lemma 6.5.2. *Given regularization constants $\gamma_1, \gamma_2 \in \mathbb{R}^+$, the solution to (6.23) is obtained as follows [117]:*

$$\left(R^{-1} + GG^T\right)\beta^{(\ell)} = R\gamma_2 c^{(\ell)}, \ell = 1,\ldots,Q, \tag{6.24}$$

where $R = \gamma_2 A - \gamma_1 V$ is a diagonal matrix and $G = [\Omega, 1_M]$. $\beta^{(\ell)} = [\beta_1^{(\ell)},\ldots,\beta_M^{(\ell)}]^T$ are the Lagrange multipliers.

Proof. The Lagrangian of the constrained optimization problem (6.23) becomes:

$$\mathcal{L}(\alpha^{(\ell)}, b^{(\ell)}, e^{(\ell)}, \beta^{(\ell)}) = \frac{1}{2}\sum_{\ell=1}^{Q}\left({\alpha^{(\ell)}}^T\alpha^{(\ell)} + (b^{(\ell)})^2\right) -$$

$$\frac{\gamma_1}{2}\sum_{\ell=1}^{Q}{e^{(\ell)}}^T V e^{(\ell)} + \frac{\gamma_2}{2}\sum_{\ell=1}^{Q}(e^{(\ell)} - c^{(\ell)})^T A(e^{(\ell)} - c^{(\ell)}) +$$

$$\sum_{\ell=1}^{Q}{\beta^{(\ell)}}^T\left(e^{(\ell)} - \Omega\alpha^{(\ell)} - b^{(\ell)}1_M\right),$$

where $\beta^{(\ell)}$ is the vector of Lagrange multipliers. Then the Karush-Kuhn-Tucker (KKT) optimality conditions are as follows,

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial e^{(\ell)}} = 0 \rightarrow e^{(\ell)} = R^{-1}\left(\gamma_2 A\, c^{(\ell)} - \beta_\ell\right),\ \ell = 1,\ldots,Q, \\ \frac{\partial\mathcal{L}}{\partial b^{(\ell)}} = 0 \rightarrow b^{(\ell)} = 1_M^T\beta^{(\ell)},\ \ell = 1,\ldots,Q, \\ \frac{\partial\mathcal{L}}{\partial \alpha^{(\ell)}} = 0 \rightarrow \alpha^{(\ell)} = \Omega^T\beta^{(\ell)}, \ell = 1,\ldots,Q, \\ \frac{\partial\mathcal{L}}{\partial \beta^{(\ell)}} = 0 \rightarrow \Omega\alpha^{(\ell)} + b^{(\ell)}1_M = e^{(\ell)},\ \ell = 1,\ldots,Q, \end{cases} \tag{6.25}$$

where R is defined as previously. Elimination of the primal variables $\alpha^{(\ell)}, e^{(\ell)}$, results in the following equation

$$\left(R^{-1} + GG^T\right)\beta^{(\ell)} = R\gamma_2 c^{(\ell)}, \ell = 1,\ldots,Q, \tag{6.26}$$

with G defined as previously. □

Obviously for large scale data, still matrix G is of size $M \times M$ which is problematic. Therefore here the reduced kernel technique can be used to overcome this issue by reducing the $M \times M$ dimensionality of kernel Ω to a much smaller dimensionality of a rectangular kernel matrix $\bar{\Omega} \in \mathbb{R}^{M\times\bar{n}}$ with $\bar{\Omega}_{ij} = K(x_i, x_j)$ and $x_i \in X$ and $x_j \in \bar{X}$. Here $\bar{X}$ is a $(\bar{n}\times d)$ random submatrix of X. Here, the subset is selected using a Rényi entropy based criterion [64]). If one works with the reduced kernel $\bar{\Omega}$ in the primal optimization problem (6.23), then by using the Sherman-Morrison-Woodbury formula [66], the solution in the dual can be obtained as follows:

$$\beta^{(\ell)} = \left[I_M - R\bar{G}\left(I_{\bar{n}+1} + \bar{G}^T R\bar{G}\right)^{-1}\bar{G}^T\right]\gamma_2 c^{(\ell)}, \ell = 1,\ldots,Q, \tag{6.27}$$

where $\bar{G} = [\bar{\Omega}, 1_M] \in \mathbb{R}^{M\times(\bar{n}+1)}$ and I_M is the identity matrix. The expression (6.27) involves the inversion of a small matrix of order $(\bar{n}+1)\times(\bar{n}+1)$. After obtaining the $\beta^{(\ell)}$, the score variables evaluated at the test set $X^{\text{test}} = \{x_i\}_{i=1}^{n_{\text{test}}}$ become:

$$\begin{aligned} e_{\text{test}}^{(\ell)} &= \bar{\Omega}^{\text{test}}\, \alpha^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}} \\ &= \left[\bar{\Omega}^{\text{test}}\ \bar{\Omega}^T\right] \beta^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}},\ \ell = 1,\ldots,Q, \end{aligned} \tag{6.28}$$

where $\bar{\Omega}_{ij}^{\text{test}} = K(x_i, x_j)$ with $x_i \in X^{\text{test}}$ and $x_j \in \bar{X}$.

The decoding scheme consists of comparing the binarized score variables for test data points with the codebook $\mathcal{CB}$ and selecting the nearest codeword in terms of Hamming distance. The procedure for Reduced MSS-KSC is summarized in Algorithm 11.

Algorithm 11: Reduced MSS-KSC approach for large scale data

Input: Training data set X, labels Y, tuning parameters γ_1 and γ_2, kernel parameter (if any), test set $X^{\text{test}} = \{x_i\}_{i=1}^{n_{\text{test}}}$ and codebook $\mathcal{CB} = \{c_q\}_{q=1}^{Q}$

Output: Class membership of test data points X^{test}

1 Select a subset matrix $\bar{X} \in \mathbb{R}^{\bar{n}\times d}$ from the original training data matrix $X \in \mathbb{R}^{M\times d}$ using Rényi entropy based criterion [64]).

2 Solve the linear system (6.27) to obtain $\{\beta^{(\ell)}\}_{\ell=1}^{Q}$ and compute the bias term $\{b^{(\ell)}\}_{\ell=1}^{Q}$ using the second equation of the KKT condition (6.4).

3 Estimate the test data projections $\{e_{\text{test}}^{(\ell)}\}_{\ell=1}^{Q}$ using (6.28).

4 Binarize the test projections and form the encoding matrix $[\text{sign}(e_{\text{test}}^{(1)}),\ldots,\text{sign}(e_{\text{test}}^{(Q)})]_{n_{\text{test}}\times Q}$ for the test points (Here $e_{\text{test}}^{(\ell)} = [e_{\text{test},1}^{(\ell)},\ldots,e_{\text{test},n_{\text{test}}}^{(\ell)}]^T$).

5 $\forall i\,(i = 1,\ldots,n_{\text{test}})$, assign x_i to class q^*, where $q^* = \underset{q}{\text{argmin}}\, d_H(e_{\text{test},i}^{\ell}, c_q)$ and $d_H(\cdot,\cdot)$ is the Hamming distance.

Remark 6.5.1. *Without loss of generality, in our experiments we set $\bar{n}$ (in Algorithm 11) equal to the number of prototype vectors, i.e. m, used in Algorithm 10.*

Remark 6.5.2. *Based on the given formulations in subsections 6.5.2 and 6.5.4, the following differences between the Reduced and Fixed-size MSS-KSC can be observed:*

In the Fixed-Size MSS-KSC approach:

- *One relies on the eigen-decomposition of the kernel matrix (associated with the prototype vectors) to approximate the feature map.*
- *The solution vector $w^{(\ell)}$ obtained by Fixed-size MSS-KSC has the same dimension as the number of prototype vectors.*
- *One solves the problem in the primal.*

In the Reduced MSS-KSC approach:

- *One does not need to apply the eigen-decomposition of the kernel matrix associated with the prototype vectors to obtain the explicit feature map.*
- *The solution vector $\beta^{(\ell)}$ obtained by Reduced MSS-KSC has the same dimension as the number of training points.*
- *One solves the problem in dual.*

6.6 Experimental Results

In this section, some experimental results are presented to illustrate the applicability of the proposed semi-supervised classification and clustering approaches. We start with a toy problem and show the differences between the obtained results when semi-supervised classification and semi-supervised clustering are applied on the same data. (see Fig. 6.1 and 6.2).

The performance of the proposed algorithms is also tested on two moons and two spirals data sets which are standard benchmarks for semi-supervised learning algorithms used in the literature [42].

Next the proposed semi-supervised classification is applied to some benchmark data sets taken from the UCI machine learning repository and the performance is compared with Laplacian SVM [21] and Means3VM [97]. Afterwards, the performance of the semi-supervised clustering on image segmentation tasks has been tested and the obtained results are compared with the kernel spectral clustering algorithm [4]. Finally the application of the semi-supervised classification is also shown in community detection of real-world networks.

6.6.1 Toy problems

The performance of the proposed semi-supervised classification and clustering algorithms are shown on a synthetic data set consisting of seven well separated

Gaussians. Some labeled data points from three of them are available (see Fig. 6.1(a)). When the semi-supervised classification algorithm is used the data are grouped into three classes due the fact that the codebook used in semi-supervised classification is a static codebook and it consists of three codewords. On the other hand, in semi-supervised clustering algorithm the codebook is designed based on the solution vector of the associated linear system and is not static, i.e. the number of codewords is not fixed and is tuned. Therefore, by applying the semi-supervised clustering one is able to partition the data into seven clusters. As it can be seen from Fig. 6.1(d) and 6.2(b), the projected data points are embedded in 3 dimensional space and yet we are able to cluster them in contrast with kernel spectral clustering algorithm [4] which requires an embedding space with dimension 6 to be able to group the given data sets into 7 clusters.

We also conducted experiments on nonlinear toy problems such as two moons and two spirals and the obtained results are shown in Fig. 6.3. For two spirals data set two scenarios are tested corresponding to different positions of the labeled data point. A comparison is made with LRGA algorithm[1] proposed in [181]. The LRGA algorithm has two parameters k and λ. In these experiments the parameter k (size of the neighborhood) is set to 10 and λ is searched within $[1, 10^{16}]$ using a logarithmic scale. As Fig. 6.3 shows, for the two moons data set the results of both method are comparable. However the results of the two spirals data set indicate that our proposed algorithm is less sensitive to the position of labeled data points[2] compared to LRGA algorithm.

In these experiments, γ_2 and σ are tuned through a grid search. The range in which the search (using a logarithmic scale) is made for γ_2 and σ is shown in Fig. 6.2(c) and Fig. 6.4. From these figures, it is apparent that there exist a range of γ_2 and σ for which the value of the utilized model selection criterion is quite high on the validation set.

6.6.2 Real-life benchmark data sets

Four benchmark data sets used in the following experiments are chosen from the UCI machine learning repository [13]. The benchmark consists of Wine, Iris, Zoo and Seeds data sets. In all cases, the data points are divided using proportion 80% and 20% into training and test counterparts respectively. Then one fourth of randomly selected data points in the training set are considered to be labeled and the remaining three fourths are unlabeled. The performance of the proposed semi-supervised classification approach (MSS-KSC), is compared

[1] Available at: http://www.cs.cmu.edu/∼yiyang/LRGA_ranking.m
[2] The equivalent of the query provided by the user.

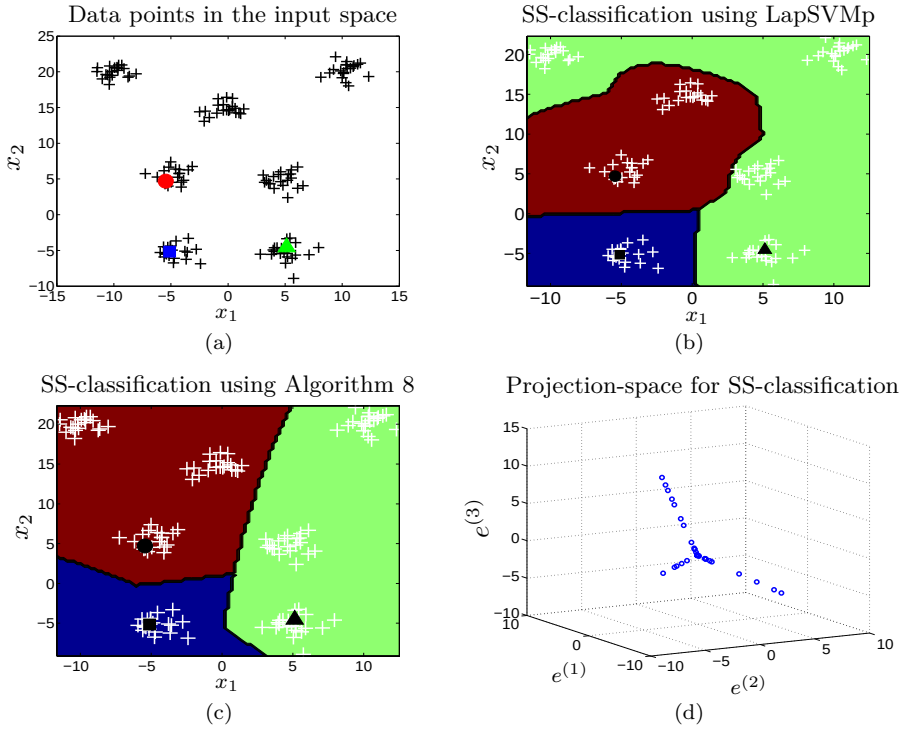


Figure 6.1: Toy Problem: Seven well separated Gaussians. The labeled data points of only three classes are available and are depicted by the blue squares (■) , green triangles (▲) and red circles (●). (a): Data points in the original space (b): Result of multi-class semi-supervised classification using LapSVMp with RBF kernel. (c): Result of the proposed multi-class semi-supervised classification with RBF kernel (Note that the algorithm detected three classes. The first class consists of one cluster whereas the second and third class consist of three clusters respectively). (d): The projections of the validation data points when the proposed semi-supervised classification algorithm is used (indicating the line structure in the projection-space).

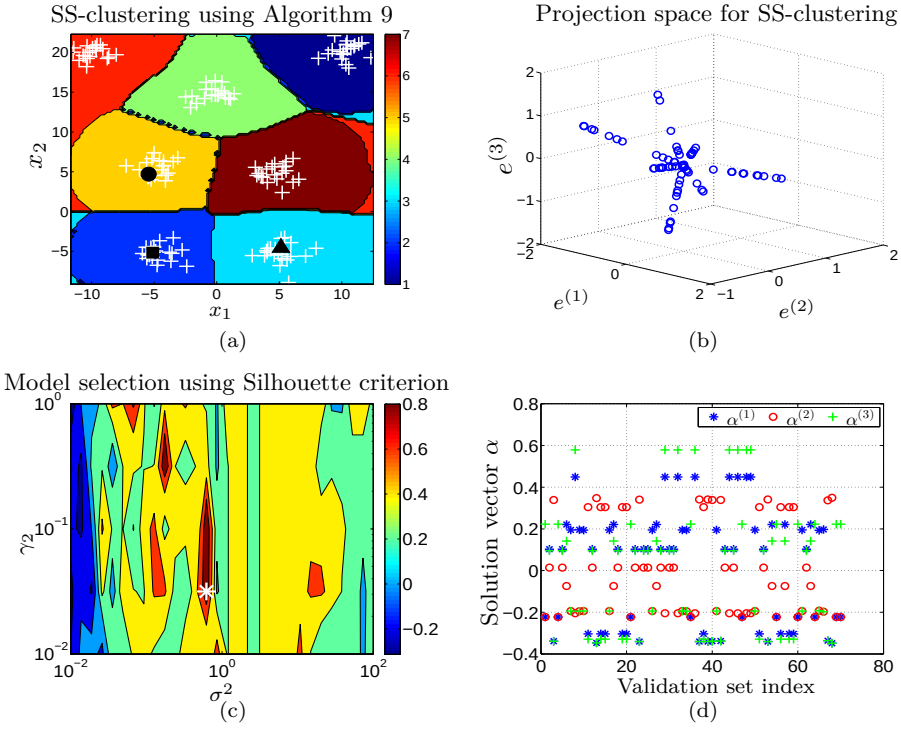


Figure 6.2: Toy Problem: Seven well separated Gaussians. The labeled data points of only three classes are available and are depicted by the blue squares (■) , green triangles (▲) and red circles (●). (a): Result of the proposed multi-class semi-supervised clustering with RBF kernel. (b): The projections of the validation data points when semi-supervised clustering algorithm is used (indicating the line structure in the projection-space). (c): Model selection for semi-supervised clustering using Silhouette validity index corresponding to the best case $T = 7$. The asterisk (*) marks the optimal model. (d): Piecewise constant property of the solution vector.

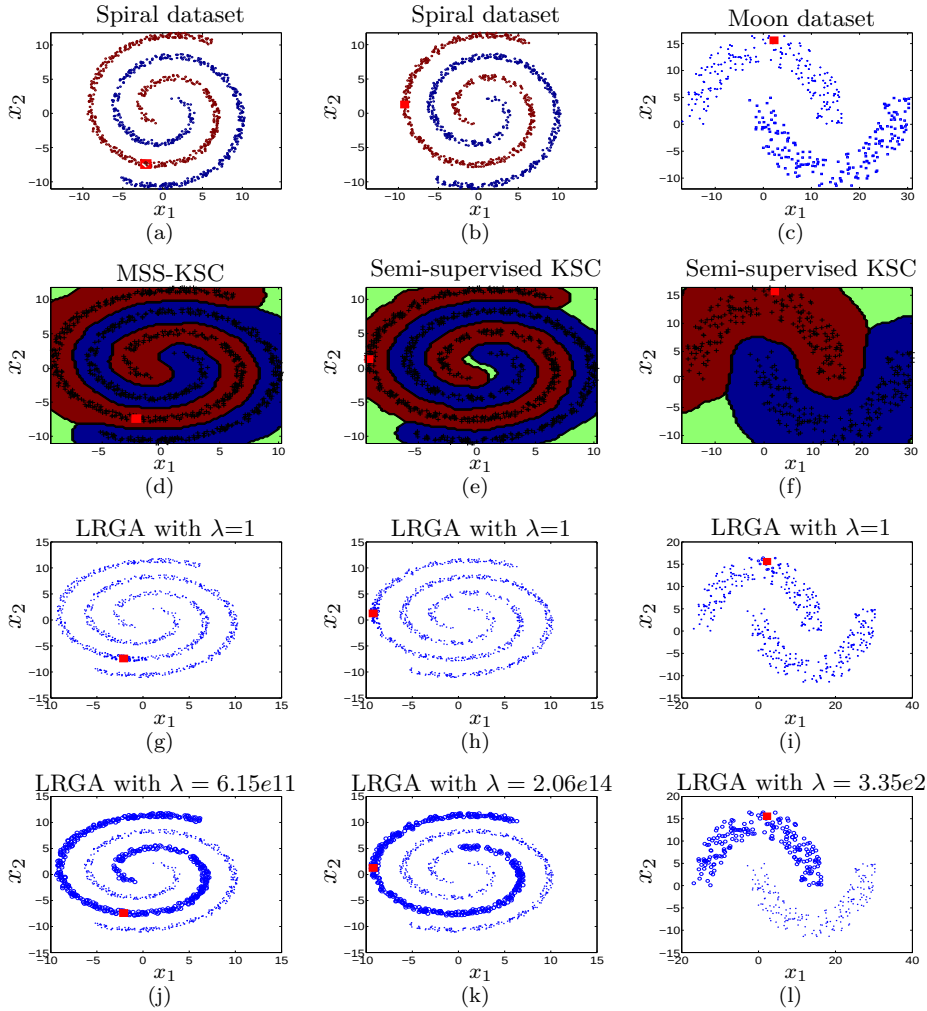


Figure 6.3: Toy Problems: two spiral and two moon data sets. The labeled data point is depicted by the red squares (■). **First row:** Data points in the original space. **Second row:** Result of the proposed semi-supervised algorithm with RBF kernel. **Third row:** Model selection of the proposed algorithm. (The asterisk (*) marks the optimal model for these examples.) **Fourth row:** Result of the LRGA algorithm corresponding to the worst case when the parameter k (size of the neighborhood) is set to 10 and λ is searched within $[1, 1e16]$ using a logarithmic scale. **Fifth row:** Result of the LRGA algorithm corresponding to the best case when the parameter k (size of the neighborhood) is set to 10 and λ is searched within $[1, 1e16]$ using a logarithmic scale.

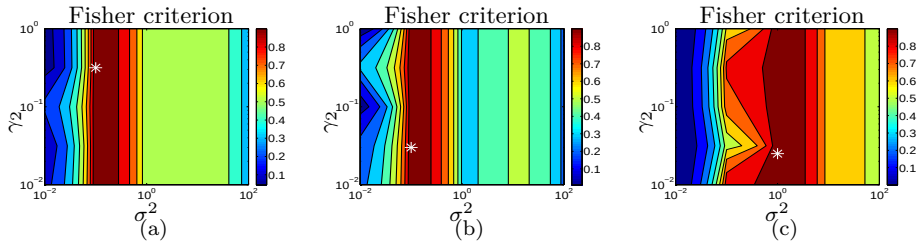


Figure 6.4: Toy Problems: two spiral and two moon data sets. The labeled data point is depicted by the red squares (■). Model selection of the proposed algorithm. (The asterisk (*) marks the optimal model for these examples.)

with Laplacian SVM (LapSVMp)[3] [21] and MeanS3VM [97] using the one-vs-all strategy.

In this experiment, the procedure used for model selection is a two-step procedure which consists of Coupled Simulated Annealing [179] initialized with random sets of parameters for the first step and the simplex method [126] for the second step. After CSA converges to some local minima, the parameters that obtained the lowest misclassification error are used for initialization of the simplex procedure to refine our selection. At every iteration for CSA method a 10-fold cross-validation is utilized. In all the experiments the RBF kernel is used.

For MeanS3VM method, the regularization parameters $C1$ and $C2$ are fixed to 1 and 0.1 (default values), respectively and the width parameter in RBF kernel is tuned with respect to the accuracy on the validation set. For the Laplacian SVMs, the kernel parameter and γ_A are tuned with respect to the accuracy on the validation set. The remaining parameters, i.e. γ_I and NN (number of nearest neighbors), are set to their default values ($\gamma_I = 1$ and $NN = 6$).

The mean and standard deviation of the accuracy rates on test data points with respect to 10 random splits are reported in Table 6.1. Table 6.1 shows that the proposed MSS-KSC approach outperforms in most cases the other approaches on these tested problems. The effect of changing the value of the user defined parameter η, used for model selection, on the performance of the proposed algorithm with respect to 10 random splits can be seen in Fig. 6.5.

[3] Available at: http://www.dii.unisi.it/∼melacci/lapsvmp/

Table 6.1: The average accuracy and the standard deviation of the LapSVMp [21], means3vm-iter [97], means3vm-mkl [97] and the proposed MSS-KSC approach on four real data sets from UCI repository [13].

Dataset (var,cls,dp)	$\mathcal{D}_{\mathrm{L}}^{\mathrm{tr}}/\mathcal{D}_{\mathrm{UL}}^{\mathrm{tr}}/\mathcal{D}^{\mathrm{test}}$	Method			
		MSS-KSC	**LapSVMp**	**means3vm-iter**	**means3vm-mkl**
Wine (13, 3, 178)	36/107/35	$\mathbf{0.96 \pm 0.02}$	0.94 ± 0.03	0.95 ± 0.02	0.94 ± 0.07
Iris (4, 3, 150)	30/90/30	0.89 ± 0.08	0.88 ± 0.05	$\mathbf{0.90 \pm 0.03}$	0.89 ± 0.01
Zoo (16, 7, 101)	21/60/20	$\mathbf{0.93 \pm 0.05}$	0.90 ± 0.06	0.88 ± 0.02	0.89 ± 0.07
Seeds (7, 3, 210)	42/126/42	$\mathbf{0.90 \pm 0.04}$	0.89 ± 0.03	0.88 ± 0.07	0.89 ± 0.02

Note: $\mathcal{D}_{\mathrm{L}}^{\mathrm{tr}}$ and $\mathcal{D}_{\mathrm{UL}}^{\mathrm{tr}}$ denote the labeled and unlabeled training points respectively. Also triple (var,cls,dp)=(variables,classes,data points).

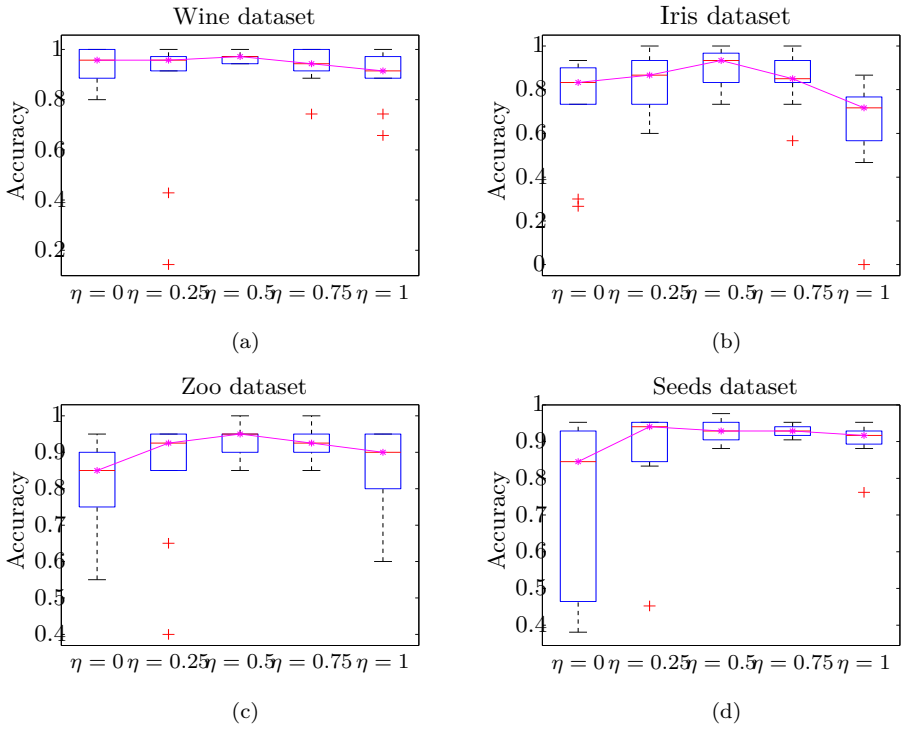


Figure 6.5: Obtained accuracy of the proposed MSS-KSC approach, with respect to different η value, over 10-simulation runs. The outliers are denoted by red "+".

6.6.3 Image segmentation

In this section, the task is to segment the given image using the proposed semi-supervised clustering. Here the aim is to show that by incorporating the side-information (labels in this case) to the unsupervised model, it is possible to improve the result of the unsupervised algorithm.

Experimental results on two synthetic images and some color images from the Berkeley image data set [104] are shown in Fig. 6.6 and 6.8. For each image, a local color histogram with a 5×5 local window around each pixel is computed using minimum variance color quantization of eight levels. A subset of 500 unlabeled pixels together with some labeled pixels (see Table 6.2) are used for training and the whole image for testing.

For the synthetic images a qualitative evaluation of both approaches is provided, since the ground truth of these images were not available. For the Berkeley images data set for which the ground truth segmentations are known, the segmentations obtained by MSS-KSC and KSC are compared with the ground truth in Table 6.2. Two evaluation criteria are used:

- F-measure, i.e. $\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$ with respect to human ground-truth boundaries.
- Variation of information (VI): it measures the distance between two segmentations in terms of their average conditional entropy. Low values indicate good match between the segmentations [7].

In these experiments, the range in which the search (using a logarithmic scale) is made for tuning the parameters γ_2 and σ are $[0, 1]$ and $[10^{-3}, 10^{1}]$ respectively. The length of the codebook p is also tuned on the interval $[Q, 2^Q]$. The score variables obtained by the proposed MSS-KSC algorithm for two images are shown in Fig. 6.6 when Silhouette criterion is used. As it can be seen, the embedding dimension (spectral embedding) is three and yet we can detect more than four clusters from the given image. Unlike the toy example 1 for these images, due to the fact that clusters are not well separated, the line structure of the score variables is less clear. In Fig. 6.7, the maximum value of the Silhouette criterion for each p (length of the codebook) while tuning γ and σ is plotted. Therefore the predicted number of clusters is equal to p for which the Silhouette value is maximum. The obtained results are shown in Fig. 6.6 and 6.8 which reveal that incorporating the prior knowledge (labels provided by human), can potentially increase the performance in the segmentation task with respect to a genuinely unsupervised approach.

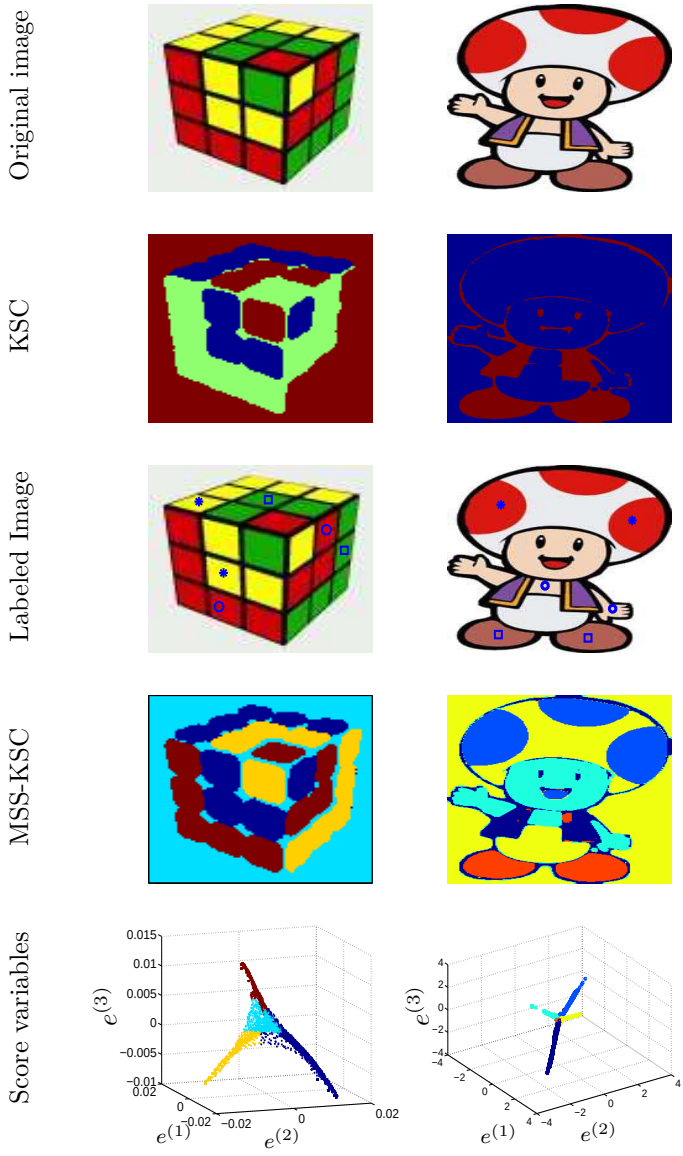


Figure 6.6: **First row:** Original image used for the KSC algorithm. **Second row:** Segmented image using the KSC approach. **Third row:** Labeled image used for the proposed MSS-KSC. **Fourth row:** Segmented image using the MSS-KSC approach. **Fifth row:** Score variables in the projection space.

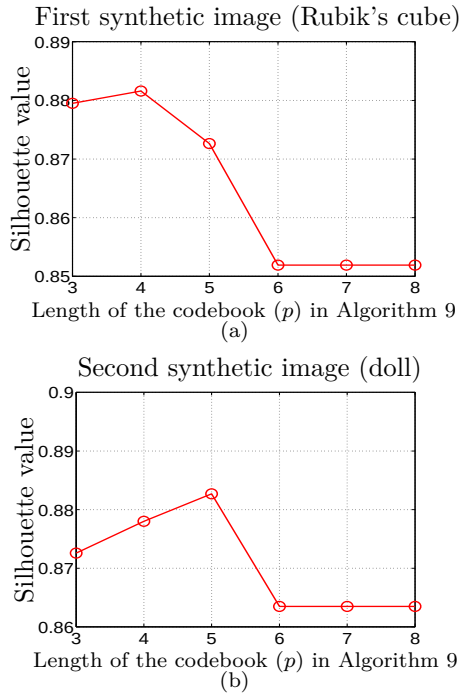


Figure 6.7: Model selection curves corresponding to the obtained Silhouette value for different p value. (a) Three labels were provided (Q=3). The maximum Silhouette value for the first syntactic image, while σ and γ_2 are tuned, over a range of $p \in \{Q, ..., 2^Q\}$. (b) Three labels were provided (Q=3). The maximum Silhouette value for the second syntactic image, while σ and γ_2 are tuned, over a range of $p \in \{Q, ..., 2^Q\}$.

Table 6.2: Comparison of KSC and MSS-KSC for image segmentations in terms of F-measure and variation of information indices

		$\mathcal{D}$		$\mathcal{D}^{val}$		**F-measure**		**Variation of information**	
Image ID	Q	$\mathcal{D}_u$	$\mathcal{D}_L$	$\mathcal{D}_u$	$\mathcal{D}_L$	**KSC**	**MSS-KSC**	**KSC**	**MSS-KSC**
100007	4	500	8	3000	8	0.57	**0.62**	**1.64**	1.95
295087	4	500	8	3000	5	0.59	**0.62**	**2.54**	2.88
372019	3	500	6	3000	6	0.40	**0.44**	2.83	**2.44**
385039	5	500	14	3000	12	0.48	**0.48**	3.20	**3.18**
388067	3	500	6	3000	6	0.60	**0.74**	4.61	**4.50**
8049	3	500	6	3000	7	0.70	**0.75**	2.22	**2.07**

Note: For variation of information the lower the value the better, whereas for F-measure the higher value the better the segmentation is.

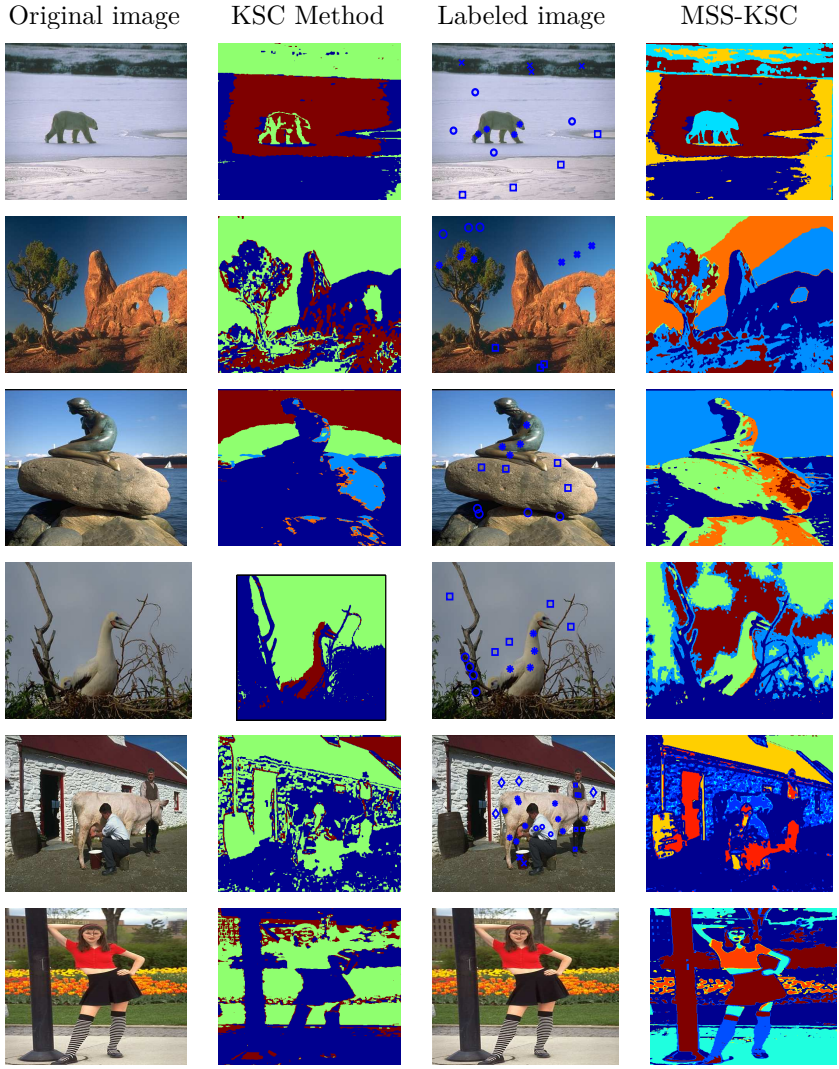


Figure 6.8: Image segmentation results using the proposed method and the KSC [4]. A subset of 500 and 3000 randomly chosen pixel histograms (unlabeled data points) together with some labeled data points are used for training and validation respectively. The whole image is used for testing. The original image is shown in the first column. The segmentation results obtained by KSC using the original images are shown in the second column. The third column shows the images labeled by human. The results of the proposed semi-supervised clustering algorithm applied on the labeled images are depicted in the fourth column.

6.6.4 Community detection

Community detection is an important topic related to complex networks [58]. It consist of finding clusters of strongly connected nodes such that nodes in the same community share more connections than nodes in different communities. Once properly identified, the community structure can help to shed light on the whole functioning of the network. Community detection is an unsupervised technique. However, if some form of prior knowledge of the community structure is present, semi-supervised techniques could in principle be used to improve the results [101, 102].

In this section the performance of the proposed method is analyzed in the community detection problems when there exist some form of prior knowledge about the existing communities. We conduct the experiments on two well known real-world networks, i.e. Karate, Football data sets shown in Fig. 6.9, which are described briefly as follows:

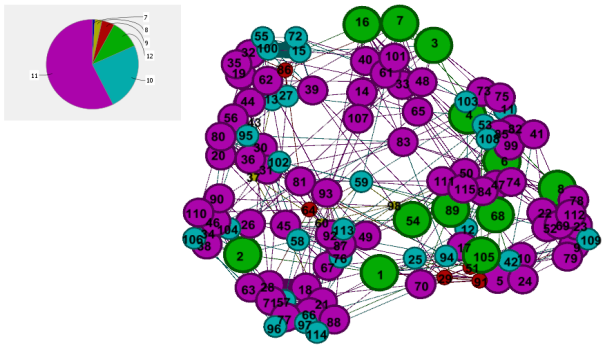
Karate: The Zachary's karate club network [182] consists of 34 member nodes, and splits into two smaller clubs after a dispute emerged during the course of Zachary's study between the administrator and the instructor.

Football: This network [65] describes American college football games and is formed by 115 nodes (the teams) and 616 edges (the games). It can be divided into 12 communities according to athletic conferences.

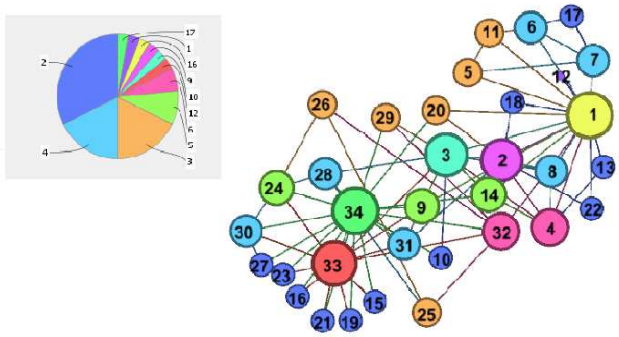
Concerning the Karate network, a comparison with the methods described in [170] is performed. In [170] a percentage of node pairs, which then are determined weather they belong to must-link or cannot-link groups, is used in the learning process. The reported results in Table 1 of [183] for different percentages of node pairs are tabulated in Table 6.3.

Since in the proposed approach we work with the labeled nodes, not pairs, we randomly select some nodes and labeled them according to the true community to which they belong. The averaged normalized mutual information (NMI) over 10 simulation runs for Karate network is reported in Table 6.3. One can observe that the proposed method is able to achieve the maximum performance using less labeled nodes than the other algorithms. In particular with 10 labeled nodes the maximum value of NMI is achieved.

Concerning the Football network, we conducted the semi-supervised classification task. The training set consists of both labeled and unlabeled nodes. 40% of each class (community) is randomly selected and form the labeled training nodes and another 40% randomly selected nodes form the unlabeled nodes. The whole network is considered as the test set and the obtained result is compard with KSC approach. The partitions found by KSC and MSS-KSC are evaluated

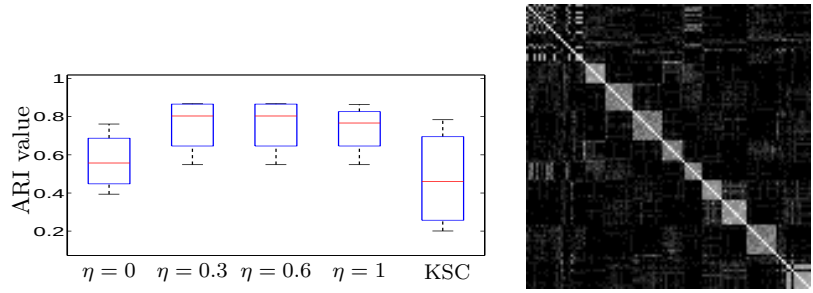


(a)



(b)

Figure 6.9: Visualization of the networks when nodes are colored according to their degree value (a) American college football undirected graph. (b) Zachary's karate club undirected graph.



(a) (b)

Figure 6.10: **American college football network**. (a) Obtained ARI value when KSC and MSS-KSC algorithm are used. (b) Kernel matrix showing the partitioning related to $\eta = 0.6$. A clear block structure revealing the presence of the 12 communities can be noticed.

according the to adjusted Rand index (ARI) [170]. The obtained ARI values on the test set after 10 runs is shown in Fig. 6.10 respectively. We can observe that, the prior knowledge incorporation helps to improve the performance with respect to KSC.

Table 6.3: **Karate network.** Comparison of MSS-KSC and methods described in [183] in terms of averaged normalized mutual information (NMI).

	Methods in [183]					**The proposed method**	
pairs constraints %	**# pairs**	[r,t]	**NMF-LSE**	**NMF-KL**	**SNMF**	**SP**	**# nodes**	**MSS-KSC**
2%	4	[4,8]	0.98	0.73	0.51	0.90	4	0.91
4%	6	[6,12]	0.99	0.85	0.60	0.96	6	0.95
5%	8	[8,16]	0.99	0.89	0.53	0.95	8	0.98
10%	16	[16,32]	1.00	0.89	0.57	1.00	10	1.00
20%	31	[31,34]	1.00	0.98	0.56	1.00	12	1.00

Note: The minimum and maximum number of nodes that could results in the given number of pairs are denoted by r and t.

6.6.5 Large scale data sets

In this section experimental results on synthetic and real-life datasets taken from UCI machine learning repository[4] [13] and LIBSVM datasets [5] [39] are given. The experiments are performed on a laptop computer with Intel Core i7 CPU and 4 GB RAM under Matlab 2012a.

[4]Available at: http://archive.ics.uci.edu/ml/datasets.html
[5]Available at: http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

The performance of the proposed FS-MSS-KSC algorithm on two moons dataset with 4000 data points is shown in Figure 6.11. The selected prototype vectors are depicted by circles. For the real datasets the size of the data on which the experiments were conducted ranges from small to large and covering both binary and multi-class classification. The amount of labeled data points used in the learning process, depending on the size of the dataset, ranges from 1% to 40% of the remaining data points (i.e. test set is not included).

Descriptions of the used datasets from [13] and [39] can be found in Table 6.4. For Ecoli and Covertype datasets we merge some of the classes in order to avoid unbalanced classes. In both Fixed-Size MSS-KSC and Reduced MSS-KSC approaches the prototype vectors (small working set) were selected via maximization of the Rényi entropy. The total amount of prototype vectors consists of prototype vectors selected from labeled and unlabeled data points. Noting that in the semi-supervised setting one usually encounters a small amount of labeled and a large amount of unlabeled data points, in our experiments, for the labeled data points the number of the prototype vectors is set as follows:

$$PV_L = \begin{cases} N_L & \text{if } N_L < 200 \\ \lceil q_1 \sqrt{N_L} \rceil & \text{otherwise,} \end{cases} \tag{6.29}$$

where $q_1 \in \mathbb{Q}^+ \backslash \{0\}$. For all the experiments q_1 is set to one. For the unlabeled data points if its number is small (less than 1000) then the number of the prototype vectors is set as follows:

$$PV_u = \begin{cases} N & \text{if } N < 500 \\ \lceil \sqrt{N} \rceil & \text{otherwise.} \end{cases} \tag{6.30}$$

In case the amount of unlabeled data points is huge, first we randomly select a fraction of them of size $N^{\text{new}} = \lceil p\, N_L \rceil$, where $p \in \mathbb{N}$, for training set and then choose the number of prototype vectors from the new set of unlabeled data points as follows:

$$PV_u = \begin{cases} \lceil N^{\text{new}} \rceil & \text{if } \lceil N^{\text{new}} \rceil < 500 \\ \lceil q_2 \sqrt{N^{\text{new}}} \rceil & \text{otherwise,} \end{cases} \tag{6.31}$$

where $q_2 \in \mathbb{Q}^+ \backslash \{0\}$. It should be noted that q_1, q_2 and p are the user defined parameters that can be designed in accordance with the available memory of the computer that is being used to conduct the experiments. The values of these parameters, i.e. q_2 and p, together with the number of training and validation data points used in the experiment are tabulated in Table 6.5. The obtained results of the proposed (Fixed-Size and Reduced) MSS-KSC approaches together with the Fixed-Size implementation of the LSSVM approach [156] are tabulated in Table 6.6. The results reported in Table 6.6, are obtained by averaging over 10 simulation runs with $\kappa = 0.25$ used in the

model selection criterion. For the LapSVMp approach, we tuned the kernel parameter and γ_A with respect to the accuracy on the validation set. The remaining parameters, i.e. γ_I and NN (the number of neighbors), are set to their default values ($\gamma_I = 1$ and $NN = 6$).

Table 6.4: Dataset statistics

Dataset	# of data points	# of attributes	# of classes
Iris	154	4	3
Spect	267	21	2
Heart	270	13	2
Ecoli	336	7	5
Pima-Indian	768	8	2
Spambase	4597	57	2
Satimage	6435	36	6
Ring	7400	20	2
Magic	19020	10	2
Cod-rna	331152	8	2
Covertype	581012	54	3

Two moons dataset with 2000 data points each

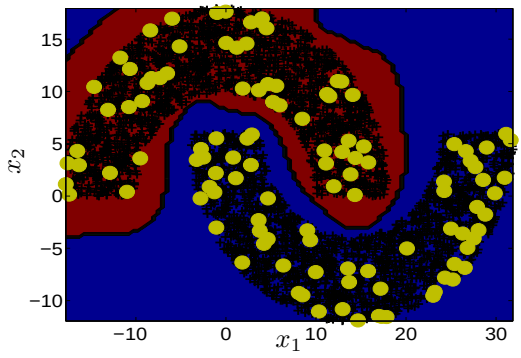


Figure 6.11: The performance of the FS-MSS-KSC method with RBF kernel on two moons dataset yielding a sparse kernel-based model. In total there are 4000 data points. The prototype vectors (small working set) selected by the Rényi entropy criterion are depicted by circles.

Table 6.6 shows that for these data one can improve the generalization performance by incorporating unlabeled data points into the learning process. It should be noted that the FS-LSSVM is a supervised algorithm that uses only the labeled training points. The training computation times for the algorithms used to obtain the results of Table 6.6 are then reported in Table 6.7. These results are expected since the FS-LSSVM does not use unlabeled data in the training process therefore it is the fastest one. The FS-MSS-KSC requires to

Table 6.5: The average test accuracy and the standard deviation of the proposed Fixed-Size, Reduced MSS-KSC approaches and Fixed-Size LSSVM [156] method on real datasets over 10 simulation runs.

Dataset	q_2/p	n_L/n_u (% of Labeled data)	$n_L^{\text{validation}}/n_u^{\text{validation}}$	PV_L/PV_u
Heart	1/1	19/76 (20%)	19/75	19/76
Pima-Indian	1/1	54/215 (20%)	54/215	54/215
Spect	1/1	19/75 (20%)	19/74	19/75
Iris	1/1	24/36 (40%)	24/36	24/36
Ecoli	1/1	54/81 (40%)	54/80	54/81
Satimage	1/1	1030/1030 (40%)	1030/1030	33/33
Ring	1/1	592/592 (20%)	592/592	25/25
Spambase	2/2	368/736 (20%)	368/736	20/55
Magic	2/2	761/1522 (10%)	761/1522	28/79
Cod-rna	1/1	1325/1325 (1%)	1325/1325	37/37
Covertype	1/1	2760/2760 (1%)	2760/2760	53/53

Note: The reported (%) of the labeled data used in the learning process, is the percentage from $\mathcal{D}\backslash\mathcal{D}^{\text{test}}$, i.e. the test set is not included.

Table 6.6: The average test accuracy and the standard deviation of the proposed Fixed-Size, Reduced MSS-KSC approaches and Fixed-Size LSSVM [156] method on real datasets over 10 simulation runs.

		Method			
Dataset	$\mathcal{D}^{\text{test}}$(%)	**FS-MSS-KSC**	**RD-MSS-KSC**	**LapSVMp**	**FS-LSSVM**
Heart	81 (30%)	$\mathbf{0.803 \pm 0.05}$	0.795 ± 0.05	0.761 ± 0.001	0.759 ± 0.05
Pima-Indian	230 (30%)	0.740 ± 0.02	0.746 ± 0.02	$\mathbf{0.748 \pm 0.001}$	0.729 ± 0.03
Spect	80 (30%)	0.832 ± 0.07	$\mathbf{0.838 \pm 0.02}$	0.821 ± 0.01	0.825 ± 0.03
Iris	30 (20%)	0.946 ± 0.05	$\mathbf{0.960 \pm 0.02}$	0.938 ± 0.13	0.601 ± 0.05
Ecoli	67 (20%)	0.746 ± 0.03	0.740 ± 0.04	$\mathbf{0.748 \pm 0.06}$	0.468 ± 0.03
Satimage	1287 (20%)	$\mathbf{0.864 \pm 0.006}$	0.831 ± 0.009	0.834 ± 0.007	0.325 ± 0.08
Ring	1480 (20%)	$\mathbf{0.975 \pm 0.005}$	0.974 ± 0.005	0.972 ± 0.006	0.968 ± 0.007
Spambase	919 (20%)	$\mathbf{0.885 \pm 0.01}$	0.883 ± 0.01	0.880 ± 0.03	0.838 ± 0.02
Magic	3804 (20%)	$\mathbf{0.836 \pm 0.006}$	0.829 ± 0.006	0.827 ± 0.005	0.825 ± 0.005
Cod-rna	66230 (20%)	$\mathbf{0.957 \pm 0.006}$	0.947 ± 0.008	0.951 ± 0.001	0.941 ± 0.006
Covertype	29050 (5%)	$\mathbf{0.715 \pm 0.005}$	0.684 ± 0.008	0.697 ± 0.001	0.362 ± 0.003

Note: The reported (%) of test set is the percentage from the entire data set.

apply an eigen-decomposition technique whereas RD-MSS-KSC does not apply any eigen-decomposition technique.

In Table 6.8, we examine the situation where the utilized size of unlabeled data is large and therefore applying LapSVMp will result in out-of-memory problems whereas the proposed FS-MSS-KSC and RD-MSS-KSC approaches that use an approximation of the feature map and reduced kernel matrix respectively, can deal with a large amount of unlabeled data points. Figure 6.12 shows the training computation times with respect to an increasing number of training points for Covertype data set. The RD-MSS-KSC showed a considerably reduced computation times due to the fact that, unlike FS-MSS-KSC, it does

Table 6.7: The average training computation times in seconds for the proposed Fixed-Size, Reduced MSS-KSC approaches, LapSVMp [21] and Fixed-Size LSSVM [156] methods on real datasets over 10 simulation runs.

	Training computation times in seconds			
Dataset	**FS-MSS-KSC**	**RD-MSS-KSC**	**LapSVMp**	**FS-LSSVM**
Heart	0.0090	0.0043	0.0267	0.0017
Pima-Indian	0.0381	0.0192	0.0295	0.0040
Spect	0.0081	0.0051	0.0265	0.0019
Iris	0.0090	0.0055	0.0025	0.0032
Ecoli	0.0395	0.0184	0.0030	0.0095
Satimage	0.1552	0.1192	0.2317	0.0277
Ring	0.0172	0.0139	0.1727	0.0069
Spambase	0.0246	0.0179	0.1497	0.0053
Magic	0.0737	0.0474	0.6026	0.0107
Cod-rna	0.3646	0.2349	7.6779	0.1590
Covertype	1.0721	0.7231	8.0201	0.6572

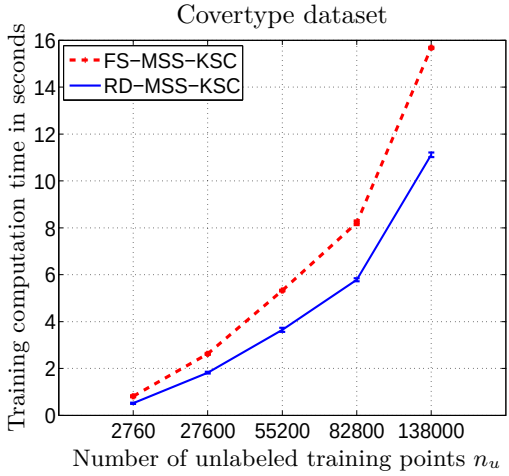


Figure 6.12: Training computation time in seconds for the Covertype dataset with an increasing number of unlabeled training points and fix number of labeled points ($n_L = 2760$). The Reduced MSS-KSC approach takes less training time than the Fixed-Size MSS-KSC approach.

not involve an eigen-decomposition step.

Table 6.8: The average test accuracy of the proposed methods on Covertype dataset. The test set is 5% of the entire dataset.

			Method		
n_L/n_u	q_2/p	PV_L/PV_u	**FS-MSS-KSC**	**RD-MSS-KSC**	**LapSVMp**
2760/2760	1/1	53/53	$\mathbf{0.715 \pm 0.01}$	0.684 ± 0.03	N.A
2760/27600	0.5/10	53/84	$\mathbf{0.729 \pm 0.04}$	0.709 ± 0.05	N.A
2760/55200	0.5/20	53/118	$\mathbf{0.731 \pm 0.02}$	0.712 ± 0.04	N.A
2760/82800	0.5/30	53/144	$\mathbf{0.739 \pm 0.04}$	0.716 ± 0.03	N.A
2760/138000	0.5/50	53/186	$\mathbf{0.742 \pm 0.05}$	0.723 ± 0.06	N.A

6.7 Conclusions

In this chapter, a multi-class semi-supervised formulation based on kernel spectral clustering (MSS-KSC) has been proposed. MSS-KSC can address both semi-supervised classification and clustering. A low embedding dimension is designed and utilized for semi-supervised clustering. The method is able to find hidden micro clusters without requiring to have a side-information about them. The validity and applicability of the MSS-KSC is shown on synthetic examples as well as on real benchmark datasets in different areas including semi-supervised classification, image segmentation and community detection problems. In order to deal with large scale data, two approaches are proposed to make the MSS-KSC scalable. The first approach, FS-MSS-KSC, uses the Nyström approximation of the feature map and solves the semi-supervised problem in the primal space. The second approach, RD-MSS-KSC, solves the problem in the dual using a reduced kernel matrix. The first approach requires an eigen-decomposition technique to obtain the explicit feature map whereas the second one does not rely on any eigen-decomposition technique.

Chapter 7

Incremental Semi-Supervised Learning Regularized by Kalman Filtering

In this chapter an on-line semi-supervised learning algorithm is formulated by a regularized kernel spectral clustering (KSC) approach. We consider the case where new data arrive sequentially but only a small fraction of it is labeled. The available labeled data act as prototypes and help to improve the performance of the algorithm to estimate the labels of the unlabeled data points. We adopt a recently proposed multi-class semi-supervised KSC based algorithm (MSS-KSC) and make it applicable for on-line data clustering. Given a few user-labeled data points the initial model is learned and then the class membership of the remaining data points in the current and subsequent time instants are estimated and propagated in an on-line fashion. The update of the memberships is carried out mainly using the out-of-sample extension property of the model. Initially the algorithm is tested on computer-generated data sets, then we show that video segmentation can be cast as a semi-supervised learning problem. Furthermore we show how the tracking capabilities of the Kalman filter can be used to provide the labels of objects in motion and thus regularizing the solution obtained by the MSS-KSC algorithm. In the experiments, we demonstrate the performance of the proposed method on synthetic data sets and real-life videos where the clusters evolve in a smooth fashion over time.

7.1 Related Work

In many real-life applications, ranging from data mining to machine perception, obtaining the labels of input data is often cumbersome and expensive. Therefore in many cases one encounters a large amount of unlabeled data while the labeled data are rare. Semi-supervised learning (SSL) is a framework in machine learning that aims at learning from both labeled and unlabeled data points [186]. SSL algorithms received a lot of attention in the last years due to rapidly increasing amounts of unlabeled data. Several semi-supervised algorithm have been proposed in the literature [21, 180, 181, 40, 174]. However, most of the SSL algorithms, operate in batch mode, hence requiring a large amount of computation time and memory to handle data streams like the ones found in real-life applications such as voice and face recognition, community detection of evolving networks and object tracking in computer vision. Therefore designing SSL algorithms that can operate in an on-line fashion is necessary for dealing with such data streams.

In the context of on-line clustering, due to the complex underlying dynamics and non-stationary behavior of real-life data, attempts have been made to design adaptive clustering algorithms. For instance, evolutionary spectral clustering based algorithms [37, 131, 46], incremental K-means [38], self-organizing time map [141]. However, in all above-mentioned algorithms the side-information (labels) is not incorporated and therefore they might underperform in certain situations. Here we adopt the recently proposed multi-class semi-supervised kernel spectral clustering (MSS-KSC) algorithm introduced in Chapter 6 and make it applicable for an on-line data clustering/classification. In MSS-KSC the core model is kernel spectral clustering (KSC) algorithm introduced in [4]. MSS-KSC is a regularized version of KSC which aims at incorporating the information of the labeled data points in the learning process. It has a systematic model selection criterion and the out-of-sample extension property.

In contrast to the methods described in [21, 180, 181, 40, 2], in the MSS-KSC approach a purely unsupervised algorithm acts as a core model and the available side information is incorporated via a regularization term. In addition, the method can be applied for both on-line semi-supervised classification and clustering and uses a low-dimensional embedding. In the MSS-KSC approach, one needs to solve a linear system of equations to obtain the model parameters. Therefore with n number of training points, the algorithm has $\mathcal{O}(n^3)$ training complexity with naive implementations. The MSS-KSC model can be trained on a subset of the data (training data points) and then applied to the rest of the data in a learning framework. Thanks to the previously learned model, the out-of-sample extension property of the MSS-KSC model allows the prediction of

the membership of a new point. However, in order to cope with non-stationary data-stream one also needs to continuously adjust the initial MSS-KSC model.

To this end, in this chapter we propose the Incremental MSS-KSC (I-MSS-KSC) algorithm which takes advantage of the available side-information to continuously adapt the initial MSS-KSC model and learn the underlying complex dynamics of the data-stream. The proposed method is rather general and can be used in several application domains including complex networks, medical imaging and video segmentation.

There have been some reports in the literature on formulating the object tracking task as a binary classification problem. For instance in [160] a tracking-based semi-supervised learning algorithm is developed for the classification of objects that have been segmented. The authors in [15] introduced a tree structured graphical model for video segmentation.

Due to the increasing demands in robotic applications, Kalman filtering has received significant attention. In particular Kalman filter has been applied in wide applications areas such as robot localization, navigation, object tracking and motion control (see [44] and references therein). The authors in [152] use the Kalman filter for monitoring a contact in a video surveillance sequence. In [185], a Kalman filter based algorithm is presented to segment the foreground objects in video sequences given non-stationary textured background. An adaptive Kalman filter algorithm has been used for video moving object tracking in [176].

In case of the video segmentation, we show how Kalman filter can be integrated into the I-MSS-KSC algorithm as a regularizer by providing an estimation of the labels throughout the whole video sequences.

7.2 Incremental Multi-class Semi-Supervised Clustering

It has been shown in Algorithm 9 of Chapter 6 that for the MSS-KSC approach, one has to solve a linear system of size n (number of training data points) in the dual to obtain the cluster membership of the data points. This is fine for batch mode but does not fit practical applications such as on-line semi-supervised clustering, in which the data are entered sequentially. If the distribution of the new arriving data points is not in line with the one of the training points, then the trained model cannot explain well the new distribution. Therefore in those cases an adaptive learning mechanism is required. In what follows we will show

how one can use the out-of-sample extension property of the MSS-KSC model for dealing with data streams in an on-line fashion.

7.2.1 Out-of-sample solution vector

In the batch MSS-KSC algorithm 9, the cluster membership of new and unseen test points $\mathcal{D}^{\text{test}} = \{x_i\}_{i=1}^{n_{\text{test}}}$ is done by an Error-Correcting Output Coding (ECOC) decoding scheme. First the cluster indicators are obtained by binarizing the score variables for test data points as follows:

$$\begin{aligned} q_{\text{test}}^{(\ell)} &= \text{sign}(e_{\text{test}}^{(\ell)}) = \text{sign}(\Phi_{\text{test}} w^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}}) \\ &= \text{sign}(\Omega_{\text{test}} \alpha^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}}),\ \ell = 1, \cdots, Q, \end{aligned}$$

where $\Phi_{\text{test}} = [\varphi(x_1), \ldots, \varphi(x_{n_{\text{test}}})]^T$ and $\Omega_{\text{test}} = \Phi_{\text{test}} \Phi^T \in \mathbb{R}^{n_{\text{test}} \times n}$, ($n$ is the number of training points). The decoding scheme consists of comparing the cluster indicators obtained in the test stage with the codebook $\mathcal{CB}$ (which is obtained in the training stage) and selecting the nearest codeword in terms of Hamming distance.

For an on-line fashion, once the model is built using the training data points, one can use the above procedure to estimate the cluster membership of the new test points. But in order for the model to be able to track the non-stationary changes in the data stream, the initial codebook $\mathcal{CB}$ should be adapted on-line so that it has the information of the more recent data points.

In addition one has to incrementally update the solution vectors α. Since in the MSS-KSC approach one needs to solve a linear system of equations, it is possible to use for instance the Sherman-Morrison-Woodbury formula [66] to efficiently update the inverse of the coefficient matrix whenever a new data point is arrived without explicitly computing the matrix inverse. In this case, also one should use some decremental algorithm to cope with non-stationary data stream [108].

Here we aim at using the out-of-sample extension capability of the MSS-KSC model. Consider n_{test} new data points, $\mathcal{D}^{\text{test}} = \{x_i\}_{i=1}^{n_{\text{test}}}$. The score variables are:

$$e_{\text{test}}^{(\ell)} = \Phi_{\text{test}} w^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}} = \Omega_{\text{test}} \alpha^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}},\ \ell = 1, \cdots, Q, \qquad (7.1)$$

where Φ_{test} and Ω_{test} are defined as previously. The third KKT condition in (6.4),

$$\alpha^{(\ell)} = (\gamma_1 V - \gamma_2 A) e^{(\ell)} + \gamma_2 c^{(\ell)}, \ell = 1, \ldots, Q,$$

links the score variables for training, i.e. e, to the solution vector α. The idea now is to extend this link to out-of-sample projections, such that we obtain an out-of-sample solution with localized properties. An estimation of the out-of-sample solution vector $\alpha^{(\ell)}, \ell = 1, \cdots, Q$ for the new test data points can be computed as follows:

$$\hat{\alpha}^{(\ell)}_{\text{test}} \stackrel{\Delta}{=} (\gamma_1 V_{\text{test}} - \gamma_2 A_{\text{test}}) e^{(\ell)}_{\text{test}} + \gamma_2 c^{(\ell)}_{\text{test}}, \quad \ell = 1, \ldots, Q, \tag{7.2}$$

where c_{test} consists of label information of some data points. $V_{\text{test}} = D^{-1}_{\text{test}} = \text{diag}(\frac{1}{d_1}, \cdots, \frac{1}{d_{n_{\text{test}}}})$ is the inverse degree matrix for the test data points. If there is no label available, one can simply estimate the solution vector by setting c_{test} and A_{test} equal zero. In case that the test dataset is sampled from the same distribution as the training data points, then the approximated out-of-sample solution vector $\hat{\alpha}_{\text{test}}$, from equation (7.2), will display localized cluster structures. Thus the data points $x_i \in \mathbb{R}^d$ are embedded in the Q-dimensional Euclidean space called α-space, i.e.

$$x_i \to \alpha_i := (\alpha^{(1)}_i, \cdots, \alpha^{(Q)}_i), \forall i = 1, \cdots, n_{\text{test}}.$$

In the MSS-KSC formulation, the clusters in the projection space (e-space) obtained by $e^{(\ell)}$ form lines with well-tuned RBF kernel parameters. Whereas the projection of the points in the α-space obtained by $\alpha^{(\ell)}$ show a localized behavior. For the sake of clarity we illustrate the projected points in both α and e-spaces, in the case of a synthetic two moons dataset in Fig 7.1.

In the case of well separated clusters, the data points that lie in the same cluster in the original space, are all mapped to one point in α-space. But in practical applications where clusters are not well separated, the data points in the same cluster in the input space will be close to each other in the α-space with respect to the other points in different clusters. Using this localized representation for out-of-sample solutions in α-space it is possible to introduce the representative or conceptual centroid of a cluster in this space.

From now on, we use two spaces: the original space $\mathcal{X}$ where the data point x_i lies and the α-space where the embedded solution vector α_i lies. Before starting to introduce the on-line semi-supervised clustering algorithm, let us introduce some definitions that will be used in the remaining of the chapter.

Definition 1. *The representative or conceptual centroid of the ith cluster $\mathcal{A}_i$ in the $\mathcal{X}$-space, is defined as the mean value of the data points in $\mathcal{A}_i$. We denote the cluster representative in the $\mathcal{X}$-space by $rep_{\mathcal{X}}(\mathcal{A}_i)$.*

Definition 2. *The representative or conceptual centroid of the ith cluster $\mathcal{A}_i$ in the α-space, is defined as the mean value of the embedded solution vector*

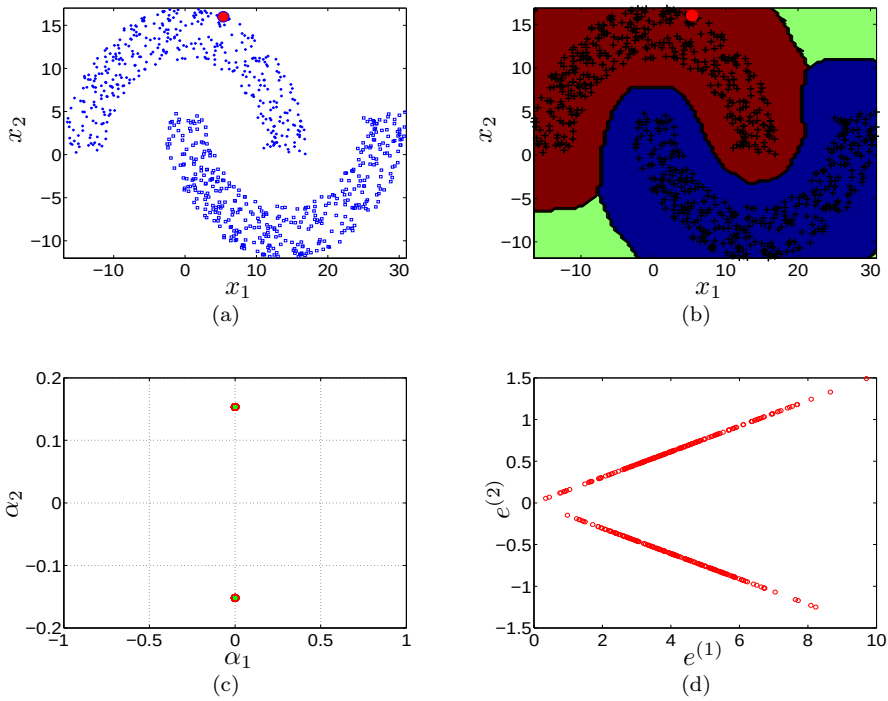


Figure 7.1: Two moons dataset: The labeled data point of only one class is available and is depicted by the red circle (•). (a): Data points in the original space. (b): The result of MSS-KSC algorithm with RBF kernel. (c): The mapped data points in the α space. (d): The mapped data points in the e space.

$\alpha_{k\in\mathcal{J}}$, $(\mathcal{J} = \{j \,|x_j \in \mathcal{A}_i\})$, *across all dimensions of the features. We denote the cluster representative in the* α*-space by* $rep_\alpha(\mathcal{A}_i)$.

Definition 3. *A prototype is defined as a point in the* $\mathcal{X}$*-space or* α*-space that has been labeled. The* j*th prototype is denoted by* $prot_{\mathcal{X},j}$ *and* $prot_{\alpha,j}$ *in* $\mathcal{X}$*-space and* α*-space respectively.*

Definition 4. *(Cluster creation and elimination). Assume that the cluster representatives* $rep_\mathcal{X}(\mathcal{A}_i(k))$ *at time step* k *are obtained. A new set of data points* $\mathcal{D}^{(k+1)}$ *at time step* $k+1$ *are defined as outliers or in other words they form a new cluster if their kernel evaluations with respect to all training data points are very close to zero. Therefore* $x_* \in \mathcal{D}^{(k+1)}$ *is considered as outlier if* $\sum_{i=1}^{n_{tr}} K(x_*, x_i)^2 < \theta_0$ *where* θ_0 *is a user defined threshold. Furthermore if*

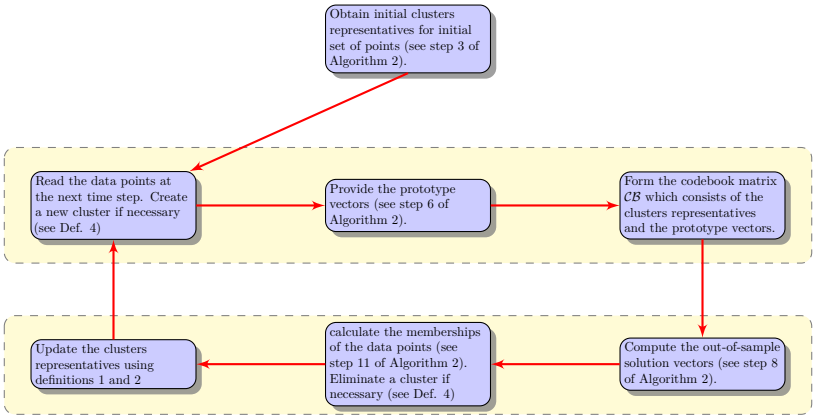


Figure 7.2: Flow-chart of the Incremental Multi-class Semi-Supervised Kernel Spectral Clustering (I-MSS-KSC) algorithm.

there is no single data point and prototype assigned to the ith cluster $\mathcal{A}_i$ then this cluster is eliminated.

In what follows, the on-line semi-supervised algorithm will be described. The proposed on-line multi-class semi-supervised clustering consists of two stages. In the first stage, one trains the MSS-KSC algorithm 9 using n training data points $\mathcal{D}$ (that contains both label and unlabeled data points) to obtain the initial solution vectors α_i and the cluster memberships. Assuming that N_c clusters are detected, the initial cluster representative $rep_{\mathcal{X}}(\mathcal{A}_i)$ and $rep_{\alpha}(\mathcal{A}_i)$ are then obtained. The aim of the second stage is to predict the membership of the new arriving data points using the updated solution vectors α_i. When batch of new data points are arrived the out-of-sample extension properties of the MSS-KSC algorithm is used to approximate the score variables associated with the new points. Next steps composed of the estimation of the projection of the points in the α-space is obtained using (7.2) and calculating the membership of the points. Finally the cluster representatives in both α and $\mathcal{X}$-spaces are updated (step 13 in Algorithm 12).

Remark 7.2.1. *If the algorithm is initialized poorly (the first stage), then one cannot expect to have a good clustering performance for the on-line stage (the second stage). The good initialization can be achieved by the aid of user labels and and well tuned model parameters. The performance of the initialization can be monitored by checking the value of an internal quality index such as Silhouette, Fisher and Davies-Bouldin indices.*

Remark 7.2.2. *The data points that are to be operated can arrive either one-by-one or as a batch of new points. In the proposed I-MSS-KSC algorithm when a batch of new data points arrives at time step k, more than one cluster can be detected without the need of using any extra step (like applying K-means in the projection space). Given Q cluster representatives at time instant $k-1$, the total number of clusters that can be created at time step k is Q. The binarized projections of the points in the alpha space is used as an indicator for the number of new clusters at time instance k. In the case of sequential one-by-one case since at time instance k, only one sample is fed to the algorithm, there will be a possibility of creation of at most one cluster.*

The proposed on-line semi-supervised clustering algorithm is summarized in Algorithm 2. The general stages of the I-MSS-KSC approach are described by the flow-chart in Fig. 7.2. In Algorithm 12, the data-stream might already have some labeled samples which then can be considered as prototypes. Otherwise, depending on the application, the prototypes can be provided by the user or for instance, for a video segmentation task the prototypes of the objects in motion can be estimated by means of a Kalman filter.

7.2.2 Computational complexity

The computational complexity of the proposed I-MSS-KSC (Algorithm 12) consists of two parts. In the first stage of the algorithm the MSS-KSC is employed to obtain the initial clusters representatives. As in MSS-KSC one needs to solve a linear system of size $n \times n$, therefore the algorithm has $\mathcal{O}(n^3)$ training complexity with naive implementations.

In the the second stage which corresponds to updating the clusters representatives for the arriving data-stream, mainly computing the kernel matrix, score variables and out-of-sample solutions vectors contribute to the complexity of the algorithm. As in the second stage, the number of training points is $n_{\text{tr}} = N_c$ (see step 7 of Algorithm 12), the overall complexity of the second stage of Algorithm 12, neglecting lower order terms, is $\mathcal{O}(n_{\text{points}} \times d \times n_{\text{tr}})$ with $n_{\text{tr}} \ll n_{\text{points}}$ and $d \ll n_{\text{points}}$. Therefore the complexity of the on-line algorithm is linear with respect to the number of data-points (n_{points}) at each time instant.

7.2.3 Regularizing I-MSS-KSC via Kalman filtering

The Kalman filter, also known as Linear Quadratic Estimator (LQE), is an algorithm that provides an efficient computational (recursive) means to

estimate the state of a linear dynamical system from noisy measurements, in a way that the variance of the estimation error is minimized.

The Kalman filter was introduced in the sixties by R. E. Kalman [80], and it has been successfully applied to the guidance, navigation and control of vehicles, particularly aircraft and spacecraft. In computer vision, the Kalman filter has been extensively used for tracking objects, and it is precisely in this context that we apply this tool in order to generate the labels for objects in motion.

As it can be seen in equation (6.1) the third term of the cost function is influenced by the labels $c^{(\ell)}$ which are provided by either the user or a Kalman filter. Therefore the Kalman filter is regularizing the solution of the MSS-KSC through $c^{(\ell)}$ values associated with the pixels of the objects in motion in a given video sequence. (See the conceptual diagram in Fig. 7.3) in equation (6.1).

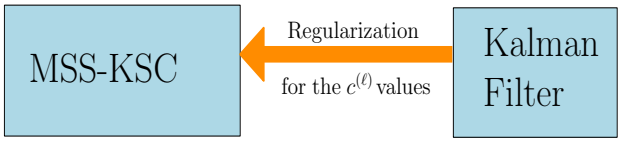


Figure 7.3: Kalman filter acts as a regularizer for the MSS-KSC algorithm

Consider the following discrete-time linear state-space model of a given dynamical system,

$$x(k+1) = Ax(k) + Bu(k) + Gw(k) \tag{7.3}$$

$$y(k) = Cx(k) + v(k)$$

where $x(k) \in \mathbb{R}^{n_x}$, $u(k) \in \mathbb{R}^{n_u}$ and $y(k) \in \mathbb{R}^{n_y}$ are the state, input and output vectors respectively, $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$ and $C \in \mathbb{R}^{n_y \times n_x}$ are the matrices defining the system dynamics, $G \in \mathbb{R}^{n_x \times n_w}$ is a weighting matrix and $w(k) \in \mathbb{R}^{n_w}$ and $v(k) \in \mathbb{R}^{n_y}$ are random variables that represent the process (model uncertainties) and measurement (measurement uncertainties) noises respectively. The process noise $w(k)$ is modeled as a Gaussian white noise with zero mean and covariance matrix $Q \in \mathbb{R}^{n_w \times n_w}$ and the measurement noise $v(k)$ is modeled as a Gaussian white noise with zero mean and covariance matrix $R \in \mathbb{R}^{n_y \times n_y}$.

Notice that for control and object tracking purposes, it is necessary to know the state vector $x(k)$. However, in general, this vector is not always available. Therefore the use of an estimator such as the Kalman filter becomes necessary in order to provide an estimate of $x(k)$ from the inputs and outputs of the system, on the basis of a mathematical model. The estimate of the state vector $x(k)$ will be denoted by $\hat{x}(k)$. For the derivation of the Kalman filter equations,

readers are referred to [19, 59]. The Kalman filter is summarized in Algorithm 13, where $P^{\mathrm{f}}(k)$ is the prior error covariance matrix, $\hat{x}(k)$ is the estimate of $x(k)$, $P(k)$ is the estimation error covariance matrix and $y(k)$ is a vector comprising the measurements.

In this work, we use some image processing techniques to roughly determine the position (measurement) of a moving object for which we would like to provide a label, and afterwards we further improve this position estimate by using a Kalman filter. We use the following kinetic model to describe the object motion:

$$\begin{aligned} s_x(k) &= s_x(k-1) + Tv_x(k-1) + \frac{T^2}{2}a_x(k-1) \\ v_x(k) &= v_x(k-1) + Ta_x(k-1) \\ s_y(k) &= s_y(k-1) + Tv_y(k-1) + \frac{T^2}{2}a_y(k-1) \\ v_y(k) &= v_y(k-1) + Ta_y(k-1) \end{aligned} \tag{7.4}$$

where T is the sampling time, $s_x(k)$, $v_x(k)$ and $a_x(k)$ are the position, velocity and acceleration of the object in the x-coordinate, and $s_y(k)$, $v_y(k)$ and $a_y(k)$ are the position, velocity and acceleration of the object in the y-coordinate. If we define the state vector as $x(k) = [s_x(k), s_y(k), v_x(k), v_y(k)]^T$, we can write down the kinematic model in a state-space form as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + Ga(k) \\ y(k) &= Cx(k) + v(k) \end{aligned} \tag{7.5}$$

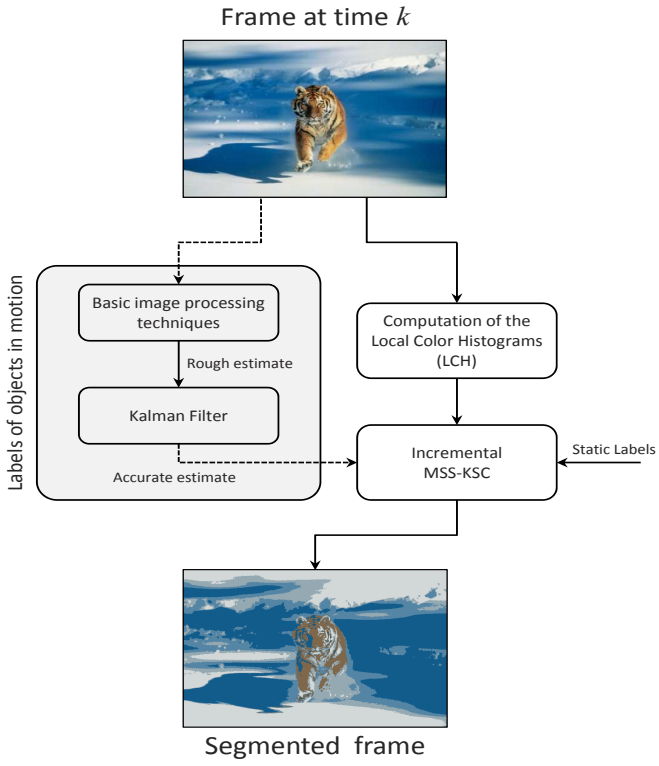
where

$$A = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, G = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

and $a = [a_x(k), a_y(k)]^T$. Here it is assumed that $a_x(k)$ and $a_y(k)$ are normally distributed, with zero mean and standard deviations σ_{a_x} and σ_{a_y} respectively. Observe that there is no $Bu(k)$ term in the previous equations given that there are no control inputs. Finally, the covariance matrices of the process and measurement noise are defined as follows:

$$Q = \begin{bmatrix} \sigma_{a_x}^2 & 0 \\ 0 & \sigma_{a_y}^2 \end{bmatrix}, R = \begin{bmatrix} \sigma_{m_x}^2 & 0 \\ 0 & \sigma_{m_y}^2 \end{bmatrix},$$

where σ_{m_x} and σ_{m_y} are the standard deviations of the measured position of the object in the x and y coordinates respectively. These measurements are



(a)

Figure 7.4: Diagram showing the interaction between Kalman filter and I-MSS-KSC approaches

generated by using some basic image processing techniques (object detection based on color, binarization, computation of centroids, etc.).

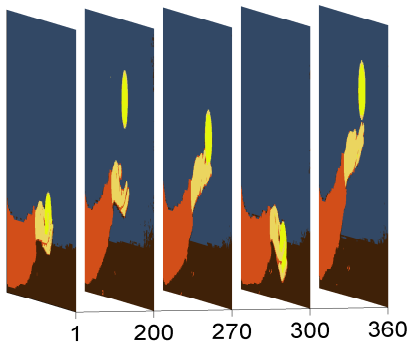


Figure 7.5: Some of the frames of the second video sequence. Each slice is treated as a batch of new data points that are fed to the algorithm.

A video sequence consists of several frames see Fig. 7.5 and each frame will be treated as batch of new data points for the algorithm.

7.3 Experimental Results

In this section, some experimental results are presented to illustrate the applicability of the proposed I-MSS-KSC algorithm. In the implementation of Algorithm 12, there are two possibilities:

- I-MSS-KSC (-): the labels (prototypes) are only provided in the first stage, i.e. just for obtaining the initial cluster representatives and the subsequent set of data points do not have any label information.
- I-MSS-KSC (+): the user can also provide the labels (prototypes) for some of the subsequent set of data points.

In order to illustrate the effect of prototypes (labels), we start with synthetic problems and show the differences between the obtained results when I-MSS-KSC(+) and I-MSS-KSC(-) are applied (see Fig. 7.6 and 7.8). Next we show the application of I-MSS-KSC requlrized by a Kalman filter to video segmentation. We used RBF kernels for all experiments unless otherwise noted.

[1] $\mathcal{A}_i(k)$ is the ith cluster at time k.
[2] Here index k denotes the kth frame.

Algorithm 12: I-MSS-KSC: On-line Semi-supervised clustering

Input: Training data set $\mathcal{D}$, labels Y, the tuning parameters $\{\gamma_i\}_{i=1}^{2}$, the kernel parameter (if any), number of clusters N_c, number of prototypes p and number of available class labels i.e. Q

Output: Cluster membership of test data points

First stage: Initialization of clusters representatives.

1 Read the training data points (initial set of points, k=1).
2 Train the MSS-KSC model using Algorithm 1 and obtain the cluster membership of the training data points.
3 Calculate the initial cluster representative $rep_{\mathcal{X}}(\mathcal{A}_i)$ and $rep_{\alpha}(\mathcal{A}_i)$ for $i = 1, \cdots, N_c$ using Definition 1 and 2.

Second stage: Updating the clusters representatives
for `k=2 to the end of the data-stream` **do**

4 Read the set of data points (n_{points}) at time k, $x_i(k), i = 1, ..., n_{\text{points}}$.
5 Detect the indices of the outlier points according to Def. 4.
6 Provide the prototypes $(prot_{\alpha,j}(k), j = 1, ..., p)$ and form the codebook matrix $\mathcal{CB}$ for the current time instant k:
$\mathcal{CB} = \left[rep_{\alpha}(\mathcal{A}_i(k))\Big|_{i=1,\ldots,N_c}, prot_{\alpha,j}(k)\Big|_{j=1,\ldots,p} \right]^T \in \mathbb{R}^{(N_c+p)\times Q}$.[1]
7 Employ the $\left[rep_{\mathcal{X}}(\mathcal{A}_i(k))\Big|_{i=1,\ldots,N_c} \right]$ as training points and calculate the score variables $e_i^{\ell}(k), i = 1, ..., n_{\text{points}}$ for $\ell = 1, ..., Q$ using (7.1).
8 Compute the out-of-sample solution vectors $\alpha_i(k), i = 1, \ldots, n_{\text{points}}$ using (7.2).
9 Form the encoding matrix for the outlier points by binarizing the obtained $\alpha_i(k)$, for all i belonging to the set of outlier indices. .
10 The unique rows of the encoding matrix obtained in step 9, indicates the number of new clusters at time step k.
11 For non-outlier points, assign $x_i(k)$ to cluster q^*, where $q^* = \underset{j}{\operatorname{argmin}}\, d_{Euc}(\alpha_i(k), \mathcal{CB}(j,:)$. Here $d_{Euc}(\cdot,\cdot)$ is the Euclidean distance and the jth row of the matrix $\mathcal{CB}$ is denoted by $\mathcal{CB}(j,:)$.
12 Eliminate a cluster if necessary according to Def. 4.
13 Update the cluster representative $rep_{\mathcal{X}}(\mathcal{A}_i(k))$ and $rep_{\alpha}(\mathcal{A}_i(k))$ according to the Definition 1 and 2.

Algorithm 13: Kalman filter

Initialization.

1 Provide the initial guess for state vector $\hat{x}(0)$ and the estimation error covariance matrix $P(0)$.

for `k=1 to end` **do**

Time update (prediction)

2 Propagate the state vector $\hat{x}(k-1)$ one-step ahead, [2]

$$\hat{x}^{\mathrm{f}}(k) = A\hat{x}(k-1) + Bu(k-1)$$

3 Propagate the covariance matrix $P(k-1)$ one-step ahead,

$$P^{\mathrm{f}}(k) = AP(k-1)A^T + GQG^T$$

Measurement update (correction)

4 Compute the Kalman gain,

$$L(k) = P^{\mathrm{f}}(k)C^T \left(CP^{\mathrm{f}}(k)C^T + R\right)^{-1}$$

5 Update $\hat{x}^{\mathrm{f}}(k)$ to $\hat{x}(k)$ by using the measurements $y(k)$,

$$\hat{x}(k) = \hat{x}^{\mathrm{f}}(k) + L(k)\left(y(k) - C\hat{x}^{\mathrm{f}}(k)\right)$$

6 Update $P^{\mathrm{f}}(k)$ to $P(k)$,

$$P(k) = \left(I - L(k)C\right)P^{\mathrm{f}}(k)$$

7.3.1 Synthetic data sets

In Fig. 7.6, there is a cloud of points which can be clustered in three groups (red, blue and green). The red and green clusters are static over time, whereas the blue cluster is moving toward the other two clusters and then it returns to its initial position.

Fig. 7.6, shows the snapshots of the evolution at specific time instants where one can see the impact of having prototypes in the incremental semi-supervised clustering. At time instants $k = 11$ and 12, where the blue cluster is close to the other two clusters, there are some points that are not correctly clustered using the I-MSS-KSC(-) algorithm. On the other hand I-MSS-KSC(+) that uses the prototypes (shown by small-squares in the Fig. 7.6) is able to cluster all the data points correctly. Hence incorporating the prototypes helps to

improve the performance. In order to evaluate the performance of the two I-MSS-KSC(-) and I-MSS-KSC(+) algorithms quantitatively, the adjusted rand index (ARI) [70] is used and the obtained results are tabulated in Table 7.1. ARI is an external evaluation criterion which measures the agreement between two partitions and takes values between zero and one. The higher the value of the ARI the better the clustering result is. In this example, at new time step k, the algorithm receives a batch of data where the number of data points is the same as that of time step $k-1$. Initially at time step $k=1$, there are 1191 data points forming three clusters. The total number of labeled data points is 21 and is fixed along all the time steps. The regularization parameters and the kernel bandwidth are $\gamma_1 = 1$, $\gamma_2 = 10^{-3}$ and $\sigma = 0.7$ respectively.

The proposed I-MSS-KSC algorithm is able to detect the creation of more than one new cluster at the given time step k, when batch of new data are fed to the algorithm. In the next example, we consider the case that three new clusters are created and eliminated at different time steps.

At time step 1, the data set consists of three clusters as in the previous example (see Fig. 7.7). Three other new clusters (clusters 4, 5 and 6) are created at time step 2. The cluster 4 and 5 are eliminated at time step 10 whereas cluster 6 disappears at time step 12. The Definition 4 is used along with the Algorithm 12 and all the above mentioned events are correctly detected. Fig. 7.7, shows the snapshots of the evolution at specific time instants where clusters are detected and eliminated. The number of data points at time step $k=1$ is 1171. In the next step 1371 new data points that form six clusters are fed to the algorithm. This number of data points is fixed until time step $k=10$ where two clusters are eliminated and therefore the total number of points is 1241 and finally at time step $k=12$ another cluster disappears from this step onward the number of data points fed into the algorithm at each step is 1171. The model parameters are $\gamma_1 = 1$, $\gamma_2 = 1$ and $\sigma = 1$ respectively.

7.3.2 Synthetic time-series

We show the applicability of the proposed I-MSS-KSC algorithm for on-line time-series clustering. The idea is to cluster signals with similar *fundamental frequencies* using a sliding window approach. Therefore we have generated two groups of signals with length 600 (each group contains 18 signals) with fundamental frequencies 0.1 rad/s and 0.3 rad/s respectively. Then from time instant $k=200$ till $k=400$, some of the pure signals of the first group are contaminated with noise which has the same fundamental frequency as the other group. The ground-truth of the time-series are shown in Fig. 7.8. For I-MSS-KSC, we have labeled one of the pure signals and a contaminated one

from the first group. The proposed I-MSS-KSC with and without labels has been applied to cluster the given time-series using a moving window approach. In this experiment the window size was set to 150. To evaluate the outcomes of the model, the average adjusted rand index (ARI) [70] is used and the results are reported in Table 7.1.

Here the similarity between the time-series is computed using the RBF kernel with the correlation distance [175]. The obtained clustering results are compared with the known ground-truth. The snapshots of the obtained results at certain time instants, where the signals from the first group have noise, are depicted in Fig. 7.9, which shows the advantage of having labels. From Fig. 7.9, one can observe that when the labels are not provided to the algorithm, it mixes things up, some of the signals from the first group are assigned to the second group and vice versa. However when the prototypes are used by the algorithm, this pattern is not observed.

Table 7.1: Averaged ARI index over time for the synthetic data points and time-series

Experiment	I-MSS-KSC(-)	I-MSS-KSC(+)
Synthetic data points	0.992	**0.999**
Synthetic time-series	0.624	**0.998**

7.3.3 Real-life video segmentation

In this section the proposed I-MSS-KSC algorithm is tested on real-life videos. We compare the performance of the proposed method with incremental K-means (IKM)[38]. K-means is one of the most popular data clustering methods due to its simplicity and computational efficiency. It works by selecting some random initial centers and then iteratively adjust the centers such that the total within cluster variance is minimized. In its incremental variant (Incremental K-means), at each time-step it uses the previous centroids to find the new cluster centers, thus avoiding to rerun the K-means algorithm from scratch [38].

The EHGB algorithm is an efficient and scalable technique for spatio-temporal segmentation of long video sequences using a hierarchical graph-based algorithm. The algorithm begins with oversegmenting a volumetric video graph into space-time regions grouped by appearance. Then a "region graph" over the obtained segmentation is constructed and this process is repeated over multiple levels to create a tree of spatio-temporal segmentations [67]. This algorithm comes with some parameters. In all the experiments, we have

selected a minimum and maximum number of regions which are stated in the corresponding caption of each of the tested video sequence. Although the EHGB algorithm does not employ labels, it is one of the state-of-the-art algorithms for video segmentation that uses past and future information (in offline mode) in order to segment the current frame. Also this algorithm uses advanced features, such as color and flow histograms. It should be noted that our algorithm uses the previous segmentation results to perform the segmentation of the current frame. And the algorithm uses only the color feature as discriminator (local color histograms).

Four real examples are used to test the validity of the proposed method. The first example shows two bouncing balls and the second example presents a human's hand throwing a ball upwards. The third video is a video sequence taken from Berkeley video segmentation dataset [3] and is called dominoes video and the fourth video is a high definition video showing birds. Descriptions of the used videos can be found in Table 7.2.

In order to extract features from a given frame, a local color histogram with a 5×5 pixels window around each pixel using minimum variance color quantization is computed. The level of quantization in general depends on the video under study. The number of levels used for each of the video is reported in Table 7.3. The χ^2 kernel $K(h^{(i)}, h^{(j)}) = \exp(-\frac{\chi^2_{ij}}{\sigma_\chi})$ with parameter $\sigma_\chi \in \mathbb{R}^+$ is used to compute the similarity between two color histograms $h^{(i)}$ and $h^{(j)}$. Here $\chi^2_{ij} = \frac{1}{2}\sum_{q=1}^{n_q} \frac{(h_q^{(i)} - h_q^{(j)})^2}{h_q^{(i)} + h_q^{(j)}}$ where n_q is the number of quantization levels.

The performance of the proposed I-MSS-KSC model depends on the choice of the tuning parameters. We set the regularization parameters $\gamma_1 = \gamma_2 = 1$ to give equal weights to unlabeled and labeled data points. The initial σ_χ (kernel parameter) is tuned using a grid search in the range $[10^{-3}, 10^1]$. The training and validation data points, i.e. $\mathcal{D}$ and $\mathcal{D}^{val}$, consist of the histograms of the chosen pixel (unlabeled data points) together with some labeled data points. These data points are used for training and validation respectively to obtain the initial cluster representatives for the first frame. Then the solution vectors and cluster representatives are updated in an on-line fashion using Algorithm 12 for the subsequent frames. The number of unlabeled/labeled training and validation data points used to obtain the initial cluster representatives are tabulated in Table 7.3. We obtain the initial model using the MSS-KSC algorithm trained on the first frame and then I-MSS-KSC is applied to segment the upcoming frames in an on-line fashion. For IKM, we let the algorithm to initialize itself and the maximum number of iterations allowed is set to 100.

[3] ftp://ftp.cs.berkeley.edu/pub/projects/vision/BVDS_train.tar.gz

Both qualitative and quantitative evaluations of the proposed approaches are provided. For quantitative evaluation of the video segmentation there is not a unique criterion to evaluate the performance of the algorithm under study. Several evaluation criteria are proposed in the literature [29, 159]. Here two criteria are used to evaluate the segmentation results. In the first criterion the segmentation obtained by the I-MSS-KSC and IKM are compared in Table 7.4 with the results of the minimum variance quantization method (the number of levels is defined by the user) [73] using the Variation of Information (VOI) index. This index measures the distance between two segmentations in terms of their average conditional entropy. Low values indicate good match between segmentations [7].

In the second criterion, the segmentations obtained by I-MSS-KSC and IKM approaches are compared in Table 7.4 with the original frames using the Cluster Quality Index (CQI) which is empirically defined in the following lines.

Suppose for a given image I, the segmented image has N_c clusters (regions). We define the quality index per cluster as follows:

$$QI_j = 1 - \frac{\sum_{i\in\{R,G,B\}} mean(|P_j^i - m_j^i|)}{3}, j = 1, \ldots, N_c,$$

where P_j^i denotes the ith channel of the RGB color for pixels of of the original image I that belong to cluster j. m_j^i is the mean value of P_j^i. Next the cluster quality index (CQI) for a given image I is heuristically defined as a weighted sum of the quality index per cluster i.e.

$$CQI(I) = \sum_{j=1}^{N_c} \theta_j QI_j, \tag{7.6}$$

where $\sum_{j=1}^{N_c} \theta_j = 1$. In our setting the highest weight is assigned to the cluster with minimum QI index. The CQI takes values in the range $[0, 1]$. The higher the value of the CQI(I) the better the segmentation is.

The obtained results of the proposed I-MSS-KSC algorithm (with two modes of implementation: I-MSS-KSC(-) and I-MSS-KSC(+)) and Incremental K-means algorithm for some of the frames of the bouncing-ball and Siamak's hand videos are depicted in Fig. 7.10 and 7.11 respectively. (The whole videos of this simulations are presented in the supplementary material of the thesis). Fig. 7.10 and 7.11, show that it is possible to improve the performance of the video segmentation by incorporating prototypes. Note that for the first video sequence, one of the ball and the table are the objects of interest. Since the table is static, the labels are provided by the user and they are fixed through out the video sequence. Whereas the ball's prototype is provided by a Kalman

filter. Here one may notice that I-MSS-KSC(+) makes it possible to improve the performance by carrying the object labeled through out the video sequence. The labeled pixels of the objects are shown by red and white asterisks (*). The obtained results of the proposed method (I-MSS-KSC(+)) and IKM for the third video are shown in Fig. 7.13. (The whole video of this simulation is provided in the supplementary material of the thesis). Fig. 7.13, indicate that the on-line segmentation results can be improved when the labels are incorporated into the algorithm. In Fig. 7.13, the labeled pixels of the objects are shown by yellow and white asterisks (*).

Table 7.2: Videos statistics

Video	width × height	# batch data points	# of frames	Frame rate
Bouncing ball	320 × 180	57600	139	29 frames/second
Siamak's hand	320 × 180	57600	395	29 frames/second
Dominoes	435 × 343	149205	121	29 frames/second
Birds	1280 × 720	921600	162	29 frames/second

Table 7.3: The number of quantization levels, unlabeled/labeled training and validation points used to obtain the initial cluster representatives.

			$\mathcal{D}$		$\mathcal{D}^{val}$	
Video	Quantization level	Q	$\mathcal{D}_u$	$\mathcal{D}_L$	$\mathcal{D}_u^{val}$	$\mathcal{D}_L^{val}$
Bouncing ball	10	3	1000	4	1500	3
Siamak's hand	8	3	800	3	1500	3
Dominoes	15	3	600	3	1500	3

Table 7.4: Comparison of IK-means, EHGB, I-MSS-KSC (-) and I-MSS-KSC (+) in terms of averaged cluster quality and variational information indices over number of frames.

	Evaluation	**Method**			
Video	Criterion	**IKM**	**EHGB**	**I-MSS-KSC (-)**	**I-MSS-KSC (+)**
Bouncing ball	CQI	0.906	0.875	0.895	**0.924**
	VOI	1.17	0.839	0.912	**0.627**
Siamak's hand	CQI	0.872	0.890	0.919	**0.925**
	VOI	1.08	1.118	0.494	**0.344**
Dominoes	CQI	0.843	**0.880**	0.855	0.866
	VOI	1.552	1.598	1.584	**1.352**
Birds	CQI	0.848	**0.874**	0.868	0.868
	VOI	0.564	0.539	**0.376**	**0.376**

Note: The higher the value of cluster quality index the better the segmentation is. The lower the value of VOI, the better the segmentation is.

7.4 Conclusions

A new incremental semi-supervised algorithm is proposed. It uses the multi-class semi-supervised kernel spectral clustering (MSS-KSC) as core model. The update of the solution vectors and the memberships are obtained using the out-of-sample solution property of the MSS-KSC approach. The user labels or labels provided by a Kalman filter are incorporated into the algorithm in an on-line fashion to improve the performance of the proposed I-MSS-KSC. The validity and applicability of the proposed method is shown on synthetic data sets and some real-life videos sequences. For the video segmentation test cases, the results obtained by the proposed I-MSS-KSC algorithm where in general better than those of the incremental K-means (IKM)[38] and comparable with the ones of the Efficient Hierarchical Graph-Based Video Segmentation (EHGB) [67].

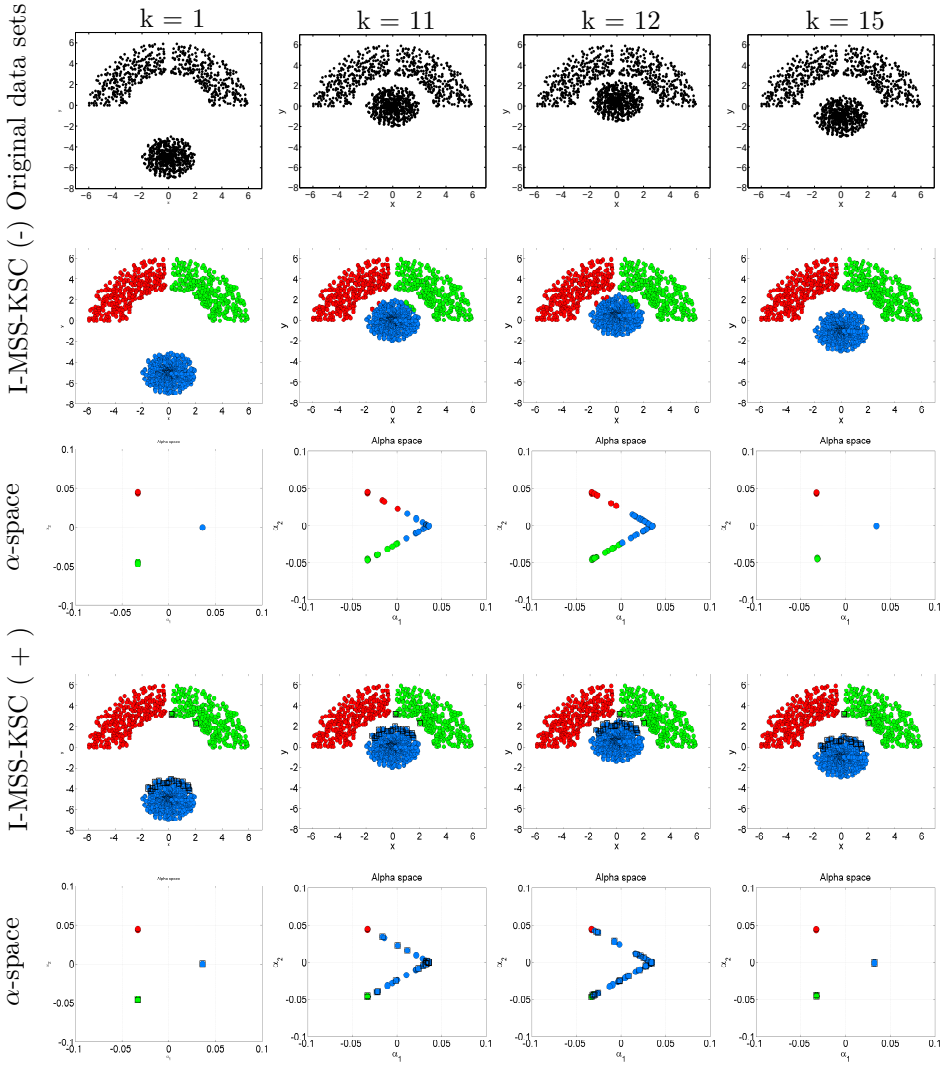


Figure 7.6: Synthetic data sets. On-line semi-supervised clustering using the proposed I-MSS-KSC approach implemented in two modes with and without prototypes (i.e. I-MSS-KSC(-) and I-MSS-KSC(+)). **First row:** The original data points at different time steps. **Second row:** I-MSS-KSC(-): The results obtained by I-MSS-KSC algorithm without the help of any prototypes after the initialization. **Third row:** The embedded solution vector α when I-MSS-KSC(-) is applied. **Fourth row:** I-MSS-KSC(+): The results obtained by the proposed I-MSS-KSC algorithm with the help of prototypes. **Fifth row:** The embedded solution vector α when I-MSS-KSC(+) is applied.

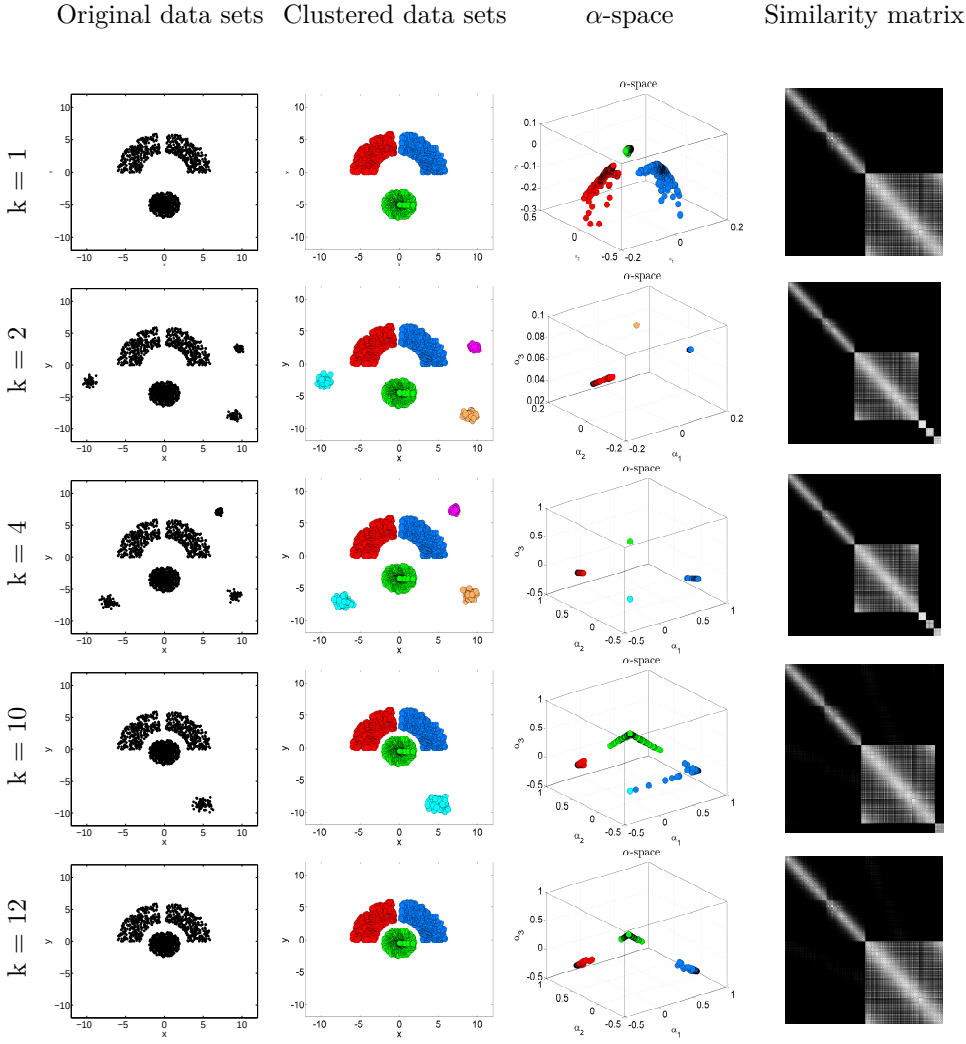


Figure 7.7: Synthetic data sets. On-line detection of the creation of more than one cluster at time step k using the proposed I-MSS-KSC(+) approach. At time step $k = 2$, three new clusters appear and evolve. Two of them disappear at time step $k = 10$ and the third one dies out at $k = 12$. The labels are just provided for the consistent clusters i.e. the ones that are always present at all the time steps and can possibly evolve over time.

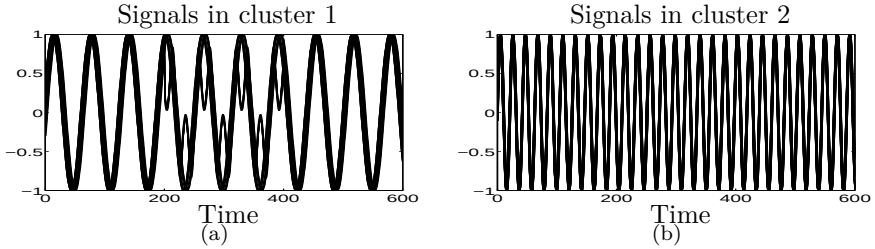


Figure 7.8: Ground-truth of the time-series (a) Signals that are in cluster 1 , (a) Signals that are in cluster 2

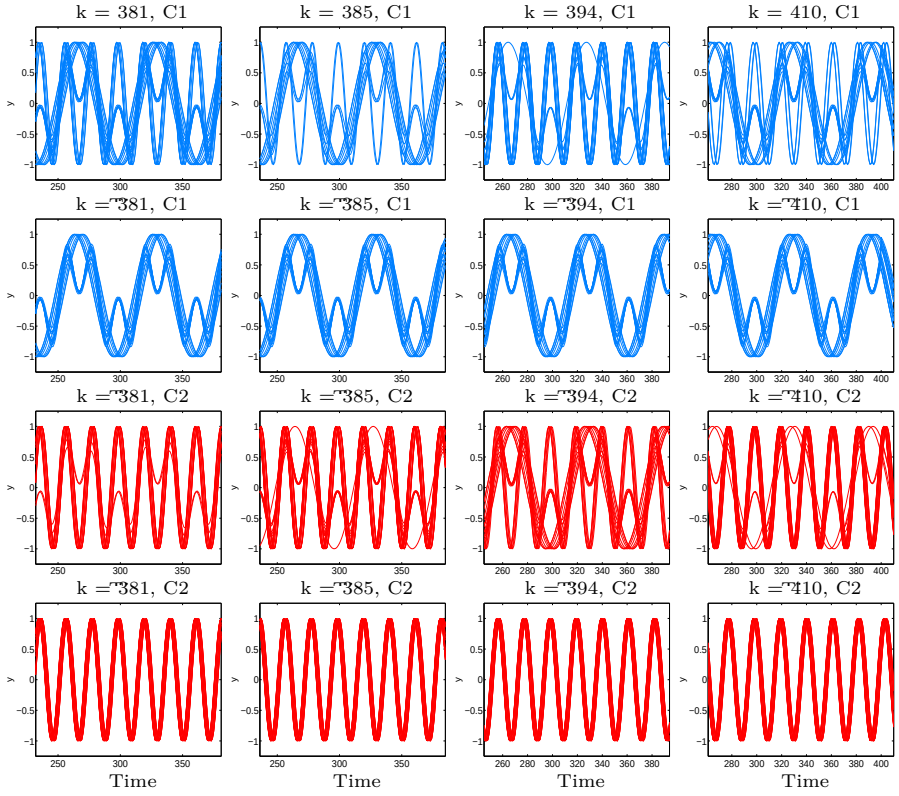


Figure 7.9: Synthetic time-series. On-line semi-supervised clustering using the proposed I-MSS-KSC approach implemented in two modes with and without prototypes (I-MSS-KSC(-) and I-MSS-KSC(+)). **First row:** I-MSS-KSC(-): The signals assigned to cluster 1 using the I-MSS-KSC algorithm without the help of any prototypes after the initialization. **Second row:** I-MSS-KSC(+): The signals assigned to cluster 1 using I-MSS-KSC algorithm with the help of the prototypes. **Third row:** I-MSS-KSC(-): The signals assigned to cluster 2 using the I-MSS-KSC algorithm without the help of any prototypes after the initialization. **Fourth row:** I-MSS-KSC(+): The signals assigned to cluster 2 using I-MSS-KSC algorithm with the help of the prototypes.

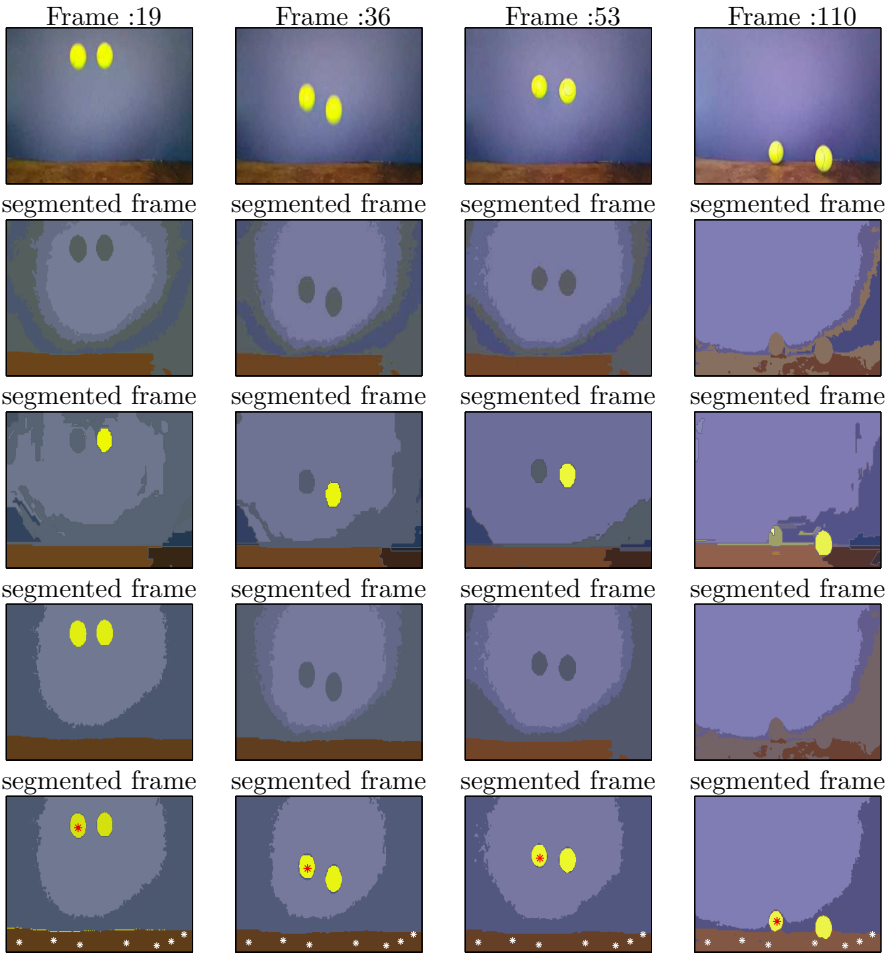


Figure 7.10: Bouncing balls video. On-line video segmentation results using the proposed I-MSS-KSC, IKM [38] and EHGB [67]. **First row:** The original frames. **Second row:** The segmentation results obtained by on-line IKM. **Third row:** The segmentation results obtained by EHGB approach [67] with Min/Max Number of regions=10/200. **Fourth row:** The segmentation results obtained by the proposed I-MSS-KSC algorithm without the help of any labeled pixels after the first frame i.e. I-MSS-KSC(-) mode. **Fifth row:** The results of the proposed I-MSS-KSC algorithm when labeled pixels for two clusters are provided during on-line segmentation, i.e. I-MSS-KSC(+) mode.

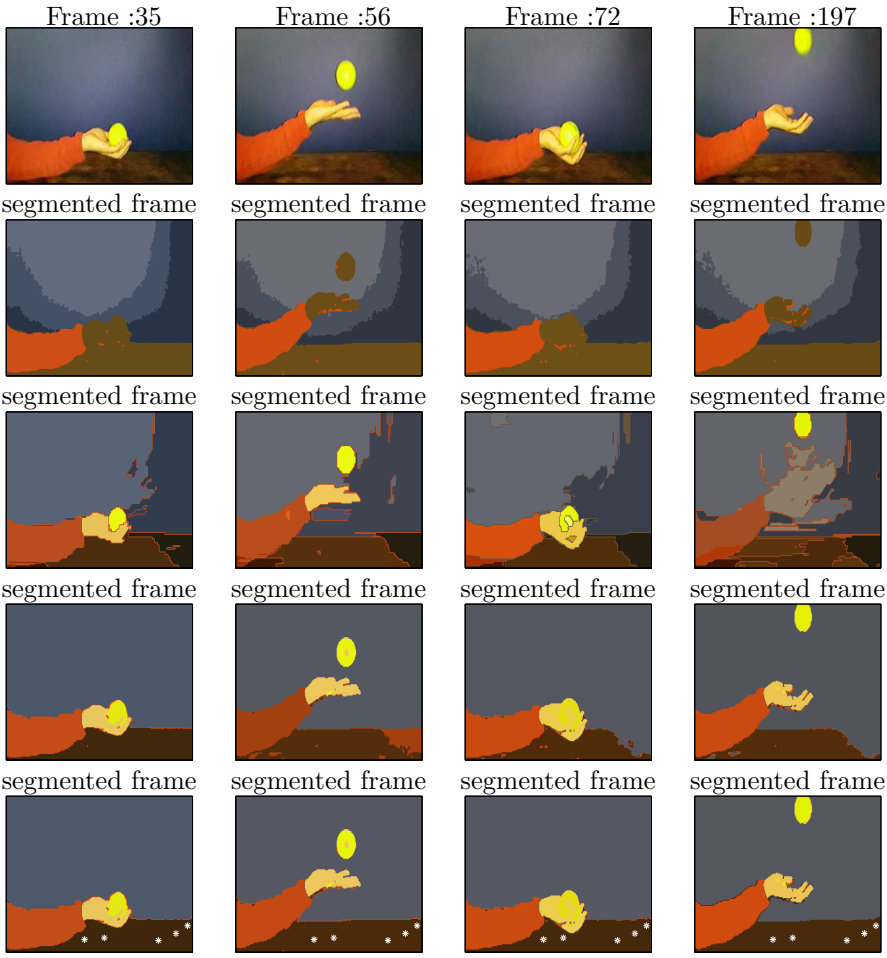


Figure 7.11: Siamak's hand video. On-line video segmentation results using the proposed I-MSS-KSC, IKM [38] and EHGB [67]. **First row:** The original frames. **Second row:** The segmentation results obtained by on-line IKM. **Third row:** The segmentation results obtained by EHGB approach [67] with Min/Max Number of regions=10/200. **Fourth row:** The segmentation results obtained by the proposed I-MSS-KSC algorithm without the help of any labeled pixels after the first frame, i.e. I-MSS-KSC(-) mode. **Fifth row:** The results of the proposed MSS-KSC algorithm when labeled pixels for two clusters are provided during on-line segmentation, i.e. I-MSS-KSC(+) mode.

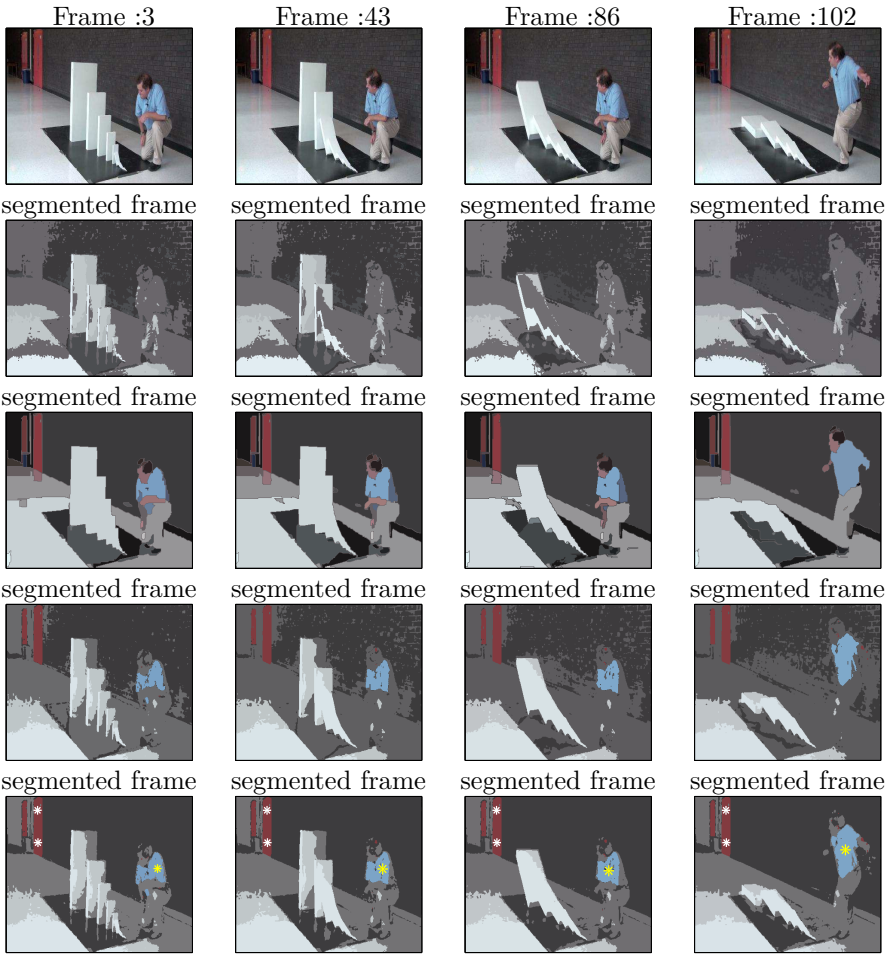


Figure 7.12: Dominoes video. On-line video segmentation results using the proposed I-MSS-KSC, IKM [38] and EHGB [67]. **First row:** The original frames. **Second row:** The segmentation results obtained by on-line K-means. **Third row:** The segmentation results obtained by EHGB approach [67] with Min/Max Number of regions=10/100. **Fourth row:** The segmentation results obtained by the proposed I-MSS-KSC algorithm without the help of any labeled pixels after the first frame i.e. I-MSS-KSC(-) mode. **Fifth row:** The results of the proposed I-MSS-KSC algorithm when labeled pixels for two clusters (objects) are provided during on-line segmentation. Note that one object is static and therefore its labels will be static and can be provided by the user.



Figure 7.13: Birds video. On-line video segmentation results using the proposed I-MSS-KSC, IKM [38] and EHGB [67]. **First row:** The original frames. **Second row:** The segmentation results obtained by on-line K-means. **Third row:** The segmentation results obtained by EHGB approach [67] with Min/Max Number of regions=10/100. **Fourth row:** The segmentation results obtained by the proposed I-MSS-KSC algorithm without the help of any labeled pixels after the first frame i.e. I-MSS-KSC(-) mode. **Fifth row:** The results of the proposed I-MSS-KSC algorithm when labeled pixels for two clusters are provided during on-line segmentation.

Chapter 8

General Conclusions

8.1 Concluding Remarks

This thesis discussed a series of novel methodologies for the incorporation of the available side-information into the kernel based core models in different contexts such as semi-supervised learning, parameter estimation of dynamical system and learning the trajectories of a dynamical system. The problems are formulated in the primal-dual setting where the additional knowledge at hand is integrated in the primal via regularization terms and/or set of constraints. The solution in the primal is in terms of the feature map and the optimal representation of the solution in the dual is obtained through the KKT optimality conditions.

This thesis adopts Least Squares Support Vector Machines (LSSVM) and Kernel Spectral Clustering (KSC) as core models and extend them, by incorporating the the prior knowledge, in the following aspects:

- **Learning the solution of a dynamical system:** The LSSVM based model is extended to learn the solution of a dynamical system governed by ordinary/partial differential equations (ODEs/PDEs) and differential algebraic equations (DAEs). The solution is learned by imposing a set of constraints into LSSVM formulation satisfying the given differential equation and its initial/boundary conditions. In the case of a linear operator, the solution in the dual is obtained by solving system of linear equations. For the nonlinear operators, one has to solve a system of nonlinear equations. The model produces a closed-form solution in the

dual. The method does not need to reduce the index of DAEs and can address both initial and boundary value problems.

- **Parameter estimation of dynamical system:** A two step approach based on LSSVM core model is introduced for the estimation of the unknown parameters (constant/time varying) of a dynamical system described by ODEs and DDEs. For the ODE case, the solution obtained by the model, is further used as an initial guess for solving the original non-convex optimization problem for estimating the unknown model parameters. For the DDE case, the unknown model parameters can be either the time delay, the history function or constant/time varying parameter presence in the model. For the parameter affine system, the problem formulation is convex otherwise non-convex.

- **Non-parallel classifiers:** A general framework for non-parallel LSSVM classifier under different loss functions is introduced. For evaluating the loss functions two types of noise are are considered: label and feature noise. The strength of each one of the loss functions are discussed under different circumstances. In particular the proposed non-parallel classifier with square loss can reduce to the LSSVM formulation for a specific regularization constant.

- **Semi-supervised learning for realistic and large scale data:** A multi-class semi-supervised learning algorithm is proposed. The kernel spectral clustering (KSC) is used as a core model and the available side-information is incorporated into the model using a regularization term. This leads to a model (MSS-KSC) capable of addressing both multi class semi-supervised classification and clustering using a low dimensional embedding. The solution in the dual is obtained by solving a system of linear equations. Furthermore the extension of the approach for large scale data is carried out by the development of FS-MSS-KSC and RD-MSS-KSC approaches.

- **Online semi-supervised learning:** A new online semi-supervised classification/clustering algorithm (I-MSS-KSC) is introduced for analyzing non-stationary data streams. The data points can arrive in two modes: one-by-one or a batch of data points. The method is able to detect more than one new cluster at the given time step k. For the video segmentation task, a Kalman filter is used to provide the labels of the objects in motion thus regularizing the solution obtained by I-MSS-KSC.

8.2 Future Research

This thesis presented new contributions originated by crossing the borders between computational mathematics and machine learning. For future work some possible directions of research may consider the adaptation/application of the proposed algorithms or the development of new methodologies based on the existing ones:

- A natural extension of the method proposed for learning the solution of PDEs would be to consider other types of loss functions and regularization terms aiming at imposing a low rank solution or having a sparse solution for high dimensional PDEs. An explicit feature map or other types of kernels could also be suitable for certain applications. Combining the proposed method with traditional numerical solvers would be an additional direction of research. Other types of PDEs that involve more complex structure could also be considered for future direction.

- Techniques based on LSSVM are designed for estimation of parameters of dynamical system whose states are all measured. Extending the approach for estimation of unknown parameters of the partially observed models is an open research area. The adaptation of the algorithm for other types of differential equations presents another challenge.

- The non-parallel classifiers introduced in this thesis can also be extended for semi-supervised learning where one encounters few labeled data points and large amount of unlabeled data points. This requires the design of a more generic model selection criteria. Addressing the scalability of the proposed approach to deal with large scale data and their parallel implementation can also be an interesting area of research.

- Developing robust to noise models for the proposed semi-supervised formulation can potentially boost the performance of the method. Although two algorithms are proposed to handle large scale data, sparse models with interpretable results and applicable for broader applications are desirable. The proposed semi-supervised algorithm in this thesis has a shallow architecture. A general class of semi-supervised learning algorithm with deep architecture having several levels of nonlinearities can also be developed.

Appendix A

Appendix

A.1 Symbolic Computing of LSSVM based models

A software tool SYM-LSSVM-SOLVER *has been written in Maple to derive the dual system and the dual model representation of LSSVM based models with equality constraints, symbolically.* SYM-LSSVM-SOLVER *constructs the Lagrangian from the given objective function and list of constraints [113]. Afterwards it obtains the KKT (Karush-Kuhn-Tucker) optimality conditions and finally formulates a linear system in terms of the dual variables. The effectiveness of the developed solver is illustrated by applying it to a variety of problems involving LSSVM based models.*

A.2 Motivation

LSSVM core models are formulated in the primal in terms of high-dimensional feature maps, equality constraints and an L_2 loss function. In most cases, solving the primal problem directly is not possible due to the high dimensionality of the variables involved in the optimization problem. Through the constrained optimization framework, it is possible to obtain a dual system where the problem is recast in terms of kernel evaluations (the so-called kernel trick) and which grows with the number of data points. Building the dual is a systematic process: first write the Lagrangian, then obtain the Karush-Kuhn-Tucker (KKT) optimality conditions and finally wrap up and formulate a system in terms of the dual variables that fulfills all KKT conditions. Fig.

A.1 shows an illustration of building models based upon LSSVM core models; as outlined in [155].

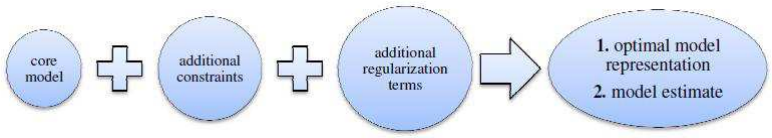


Figure A.1: Illustration of advanced LSSVM models.

A.3 Development of Symbolic Solver

In order to be able to work with a symbolic solver for LSSVM model, at first the model should be transformed to the symbolic expressions i.e. in the matrix or vector notation. It should be noted that this stage is done by the user before utilizing the symbolic solver. An example is provided to clarify this procedure.

Let us consider a given training set $\{x_i, y_i\}_{i=1}^{N}$ with input data $x_i \in \mathbb{R}^d$ and output data $y_i \in \{-1, 1\}$. The LSSVM model for classification [155], can be rewritten in a matrix form as follows

$$\begin{aligned} \underset{w,b,e}{\text{minimize}} \quad & \frac{1}{2} w^T w + \frac{\gamma}{2} e^T e \\ \text{subject to} \quad & Y \left[\Phi w + b 1_N \right] = 1_N - e \end{aligned} \tag{A.1}$$

where $\gamma \in \mathbb{R}^+, b \in \mathbb{R}$, $e \in \mathbb{R}^N$, $w \in \mathbb{R}^h$, $Y = \text{diag}(y_1, y_2, \ldots, y_N) \in \mathbb{R}^{N \times N}$, $1_N \in \mathbb{R}^N$, $\Phi \in \mathbb{R}^{N \times h}$ with

$$\Phi = \left[\phi(x_1) \cdots \phi(x_N) \right]^T,$$

$\phi(\cdot) : \mathbb{R}^d \longrightarrow \mathbb{R}^h$ is the feature map and h is the dimension of the feature space.

The approach on which the LSSVM symbolic solver is based can be summarized as follows:

1. Constructing the Lagrangian.
2. Taking derivatives of the Lagrangian with respect to the primal and dual variables and setting them equal zero.
3. Elimination of primal variables (or part of it).

4. Expressing the solution in terms of the Lagrange multipliers.
5. Obtaining the dual representation of the model.

A.4 *SYM-LSSVM-SOLVER* Package

A specific module, denoted by *SYM_LS_SVM_SOLVER*, is designed for the symbolic solver for LS-SVMs. This module is composed of four main procedures denoted by:

- *Pro_Lag*,
- *Pro_KKT*,
- *Pro_Dual system*,
- *Pro_Dual Model*

In Maple one read this as follows:

```
> print(SYM_LS_SVM_SOLVER);
```

module()

export Pro_Lag, Pro_KKT, Pro_Dual System, Pro_Dual Model;

end module

More details of these procedures are discussed in the following subsections.

A.4.1 Procedure *Pro-Lag*

The aim of this procedure is to form the Lagrangian from a given primal problem. The arguments of the Pro_Lag procedure are thus the objective function, list of constraints and Lagrange multipliers, respectively. It should be noticed that in our code the vectors are considered as a special case of matrices. Also the possibility that the users can define the type of the matrix is provided.

Example 1. Consider the LSSVM model (A.1). One initially reads the package into memory using the *'with'* command. A second task is to utilize the *'assume'* command to specify the matrix variables. If the variable has additional properties such as being symmetric or positive definite, the

additionally function can be used which adds additional assumptions without removing previous assumptions.

```
> with(SYM_LS_SVM_SOLVER);
> assume(w::Matrix,e::Matrix,Phi::Matrix,
> N1::Matrix,alpha::Matrix,Y::Matrix),additionally(Y::symmetric);
> L[1]:=Pro_Lag(0.5*w^T.w+0.5*gamma*(e^T.e),
> [Y.Phi.w+b*(Y.N1)=N1-e],[alpha]);
```

$\mathcal{L}_1 = 0.5\, w^T w + 0.5\, \gamma e^T e - \alpha^T \cdot Y \cdot \Phi \cdot w - b\alpha^T \cdot Y \cdot N1 + \alpha^T \cdot N1 - \alpha^T \cdot e$ Note that *N1* is a vector of all ones and equals 1_N.

Example 2. Consider the following problem,

$$\begin{aligned} \underset{w,b,e,\widehat{Y}}{\text{minimize}} \quad & \frac{1}{2} w^T w + \gamma e^T e + \eta (\widehat{Y} - Y^*)^T (\widehat{Y} - Y^*) \\ \text{subject to} \quad & Y - \widehat{Y} = e \\ & \widehat{Y} = \Phi w + b 1_N \end{aligned} \tag{A.2}$$

```
> assume(e::Matrix,w::Matrix,
> Y::Matrix,Yhat::Matrix,alpha[1]::Matrix,alpha[2]::Matrix,
> Phi::Matrix,N1::Matrix, Ystr::Matrix);
> L[2]:=Pro_Lag(0.5*(w^T.w)+gamma*(e^T.e)+eta*((Yhat-Ystr)^T.
> (Yhat-Ystr)),[Y-Yhat-e,Yhat-Phi[1].w-b*N1],[alpha[1],alpha[2]]);
```

$$\mathcal{L}_2 = \frac{1}{2} w^T w + \gamma\, e^T e + \eta\, (Yhat - Ystr)^T \cdot (Yhat - Ystr) + {\alpha_1}^T \cdot Y -$$

$${\alpha_1}^T \cdot Yhat - {\alpha_1}^T \cdot e + {\alpha_2}^T \cdot Yhat - {\alpha_2}^T \cdot \Phi \cdot w - b{\alpha_2}^T \cdot N1$$

Example 3. As another example, we consider the data visualization model, see ([154]),

```
> with(SYM_LS_SVM_SOLVER);
> assume(z::Matrix,N1::Matrix,P[D]::Matrix);
> dims:=2;
> for k from 1 to dims do
> assume(w[k]::Matrix,e[k]::Matrix,Phi[k]::Matrix,v[k]::Matrix,
> alpha[k]::Matrix,C[k]::Matrix,M[k]::Matrix,Omega[k]::Matrix,
> beta[1,k]::Matrix,e[1,k]::Matrix); end do;
```

```
> L[3]:=Pro_Lag(-0.5*gamma*z^T.z+0.5*(z-P[D].z)^T.(z-P[D].z)+
> (gamma/2)*(sum(w[j]^T.w[j],j=1..dims))+0.5*eta*(sum(e[j]^T.e[j],
> j=1..dims)),[seq(v[j]^T.z-Phi[j].w[j]-b[j]*N1=e[j],j=1..dims),
> seq(C[j]^T.z=q[j]+e[1,j],j=1..dims)],
> [seq(alpha[j],j=1..dims),seq(beta[1,j],j=1..dims)]);
```

$$\mathcal{L}_3 = -0.5\,\gamma\, z^T z + 0.5\,(z - P_D \cdot z)^T \cdot (z - P_D \cdot z) + 0.5\,\gamma\,\left({w_1}^T w_1 + {w_2}^T w_2\right) +$$
$$0.5\,\eta\,\left({e_1}^T e_1 + {e_2}^T e_2\right) + {\alpha_1}^T \cdot {v_1}^T \cdot z - {\alpha_1}^T \cdot \Phi_1 \cdot w_1 - b_1 {\alpha_1}^T \cdot N1 -$$
$${\alpha_1}^T \cdot e_1 + {\alpha_2}^T \cdot {v_2}^T \cdot z - {\alpha_2}^T \cdot \Phi_2 \cdot w_2 - b_2 {\alpha_2}^T \cdot N1 - {\alpha_2}^T \cdot e_2 + {\beta_{1,1}}^T \cdot {C_1}^T$$
$$\cdot z - {\beta_{1,1}}^T \cdot q_1 - {\beta_{1,1}}^T \cdot e_{1,1} + {\beta_{1,2}}^T \cdot {C_2}^T \cdot z - {\beta_{1,2}}^T \cdot q_2 - {\beta_{1,2}}^T \cdot e_{1,2}$$

A.4.2 Procedure *Pro-KKT*

After obtaining the Lagrangian, the task is to take derivatives of this function with respect to the primal variables and Lagrange multipliers. In our code, Procedure *Pro-KKT* sets the derivatives of the Lagrangian to zero which leads to the system of linear equations.

The built-in differentiator in Maple (i.e *diff* command) is not able to handle the derivative with respect to a vector or matrix (of known dimension, but unknown values). Therefore a special procedure so called *Pro_DIFF* is designed to do differential operations on generalized matrices symbolically, under the framework of LS-SVMs. *Pro_DIFF* has two parameters, the algebraic expression that has to be differentiated and differentiation variable respectively.

Most cases encountered when solving LS-SVMs are as follows,

$$\frac{\partial X^T A}{\partial X} = \frac{\partial A^T X}{\partial X} = A, \quad \frac{\partial A^T X B}{\partial X} = AB^T, \quad \frac{\partial X^T X}{\partial X} = 2X, \quad \frac{\partial X^T A X}{\partial X} = (A + A^T)X$$

Where A, B, X are symbols for matrices. For more details we refer to [134].

Let us give an example to show how this procedure works individually,

```
> with(SYM_LS_SVM_SOLVER);
> assume(A::Matrix,B::Matrix,X::Matrix),additionally(A::symmetric);
> Pro_DIFF(b*(X^T.A.X)*q+X^T.X-A^T.X^T.B,X);
```

$$2b(A.X)q + 2X - BA^T$$

Note that b and q were not defined as matrix, so they just behave like a scalar.

Having the Lagrangian function available from the *Pro_Lag*, we can call the function *Pro_KKT* to generate the KKT optimality conditions. The parameters of *Pro_KKT* are thus the Lagrangian, list of differentiation variables and number of w vectors (the dimension of the problem) respectively. In order to illustrate the procedure we apply it to the example 2 and 3, thus the KKT optimality conditions are as follows,

For example 2,

```
> Pro_KKT(L[2],[w,e,alpha[1],alpha[2],Yhat,b],1);
```

$$\frac{\partial L_2}{\partial w} = 2\,w - \Phi^T \cdot \alpha_2 = 0,$$

$$\frac{\partial L_2}{\partial e} = 2\,\gamma\,e - \alpha_1 = 0,$$

$$\frac{\partial L_2}{\partial \alpha[1]} = Y - Yhat - e = 0,$$

$$\frac{\partial L_2}{\partial \alpha[2]} = Yhat - \Phi \cdot w - bN1 = 0,$$

$$\frac{\partial L_2}{\partial \widehat{Y}} = 2\,\eta\,Yhat - 2\,\eta\,Ystr - \alpha_1 + \alpha_2 = 0,$$

$$\frac{\partial L_2}{\partial b} = -N1^T \cdot \alpha_2 = 0.$$

For example 3,

```
> Pro_KKT(L[3],[seq(w[i],i=1..dims),seq(e[i],i=1..dims),
> seq(e[1,i],i=1..dims),seq(alpha[i],i=1..dims),
> seq(beta[1,i],i=1..dims),seq(b[i],i=1..dims),z],2);
```

$$\frac{\partial L_3}{\partial w_1} = \gamma\, w_1 - {\Phi_1}^T \cdot \alpha_1 = 0,$$

$$\frac{\partial L_3}{\partial w_2} = \gamma\, w_2 - {\Phi_2}^T \cdot \alpha_2 = 0,$$

$$\frac{\partial L_3}{\partial e_1} = 1.0\, \eta\, e_1 - \alpha_1 = 0,$$

$$\frac{\partial L_3}{\partial e_2} = 1.0\, \eta\, e_2 - \alpha_2 = 0,$$

$$\frac{\partial L_3}{\partial e_{1,1}} = -\beta_{1,1} + 1.0\, \eta\, e_{1,1} = 0$$

$$\frac{\partial L_3}{\partial e_{1,2}} = -\beta_{1,2} + 1.0\, \eta\, e_{1,2} = 0,$$

$$\frac{\partial L_3}{\partial \alpha_1} = {v_1}^T \cdot z - \Phi_1 \cdot w_1 - b_1 N1 - e_1 = 0,$$

$$\frac{\partial L_3}{\partial \alpha_2} = {v_2}^T \cdot z - \Phi_2 \cdot w_2 - b_2 N1 - e_2 = 0,$$

$$\frac{\partial L_3}{\partial \beta_{1,1}} = {C_1}^T \cdot z - q_1 - e_{1,1} = 0,$$

$$\frac{\partial L_3}{\partial \beta_{1,2}} = {C_2}^T \cdot z - q_2 - e_{1,2} = 0,$$

$$\frac{\partial L_3}{\partial b_1} = -{N1}^T \cdot \alpha_1 = 0,$$

$$\frac{\partial L_3}{\partial z} = -1.0\, \gamma\, z + 1.0\ {(I - P_D)}^T \cdot (I - P_D) \cdot z + v_1 \cdot \alpha_1 + v_2 \cdot \alpha_2 + C_1 \cdot \beta_{1,1} +$$

$$C_2 \cdot \beta_{1,2} = 0,$$

$$\frac{\partial L_3}{\partial b_2} = -{N1}^T \cdot \alpha_2 = 0.$$

A.4.3 Procedure *Pro-Dual System*

The procedure *Pro-Dual System*, as its name suggests, will produce the corresponding dual system for the given primal problem. The remaining variables are defined by the user. *Pro-Dual System* has four parameters, Lagrangian, differentiation variables, remaining variables and number of w vectors, respectively. In what follows, we illustrate this procedure by applying it to the example 2.

For example 2, we have

```
> Pro_Dualsystem(L[2],[w,e,alpha[1],alpha[2],Yhat,b],[alpha[2],b],1);
```

$$G1 \cdot \begin{bmatrix} \alpha_2 \\ b \end{bmatrix} = \begin{bmatrix} 2\,Y\gamma + 2\,\eta\; Ystr \\ 0 \end{bmatrix},$$

$$\texttt{`where G1`} = \begin{bmatrix} -\gamma\,\Omega \cdot I_N - I_N - \eta\,\Omega \cdot I_N & 2\,\gamma\, I_N N1 + 2\,\eta\, I_N N1 \\ N1^T \cdot I_N & 0 \end{bmatrix}$$

where Ω=$\Phi\Phi^T$ denotes the $N \times N$ kernel matrix.

For example 3, we have

```
> Pro_Dual system(L[3],[seq(w[i],i=1..dims),seq(e[i],i=1..dims),
> seq(e[1,i],i=1..dims),seq(alpha[i],i=1..dims),seq(beta[1,i],i=1..dims)
> seq(b[i],i=1..dims),z],
> [z,b[1],b[2]],2);
```

$$G1 \cdot \begin{bmatrix} z \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} C_1 \cdot q_1\eta + C_2 \cdot q_2\eta \\ 0 \\ 0 \end{bmatrix}$$

$$\texttt{`where G1`} = \begin{bmatrix} U & -v_1 \cdot {M_1}^{-1} \cdot I_N N1 & -v_2 \cdot {M_2}^{-1} \cdot I_N N1 \\ -N1^T \cdot {M_1}^{-1} \cdot {v_1}^T \cdot I_N & N1^T \cdot {M_1}^{-1} \cdot I_N N1 & 0 \\ -N1^T \cdot {M_2}^{-1} \cdot {v_2}^T \cdot I_N & 0 & N1^T \cdot {M_2}^{-1} \cdot I_N N1 \end{bmatrix}$$

$$\texttt{`where U`} = -1.0\,\gamma\, I_N + 1.0\,(I - P_D)^T \cdot (I - P_D) \cdot I_N + v_1 \cdot {M_1}^{-1} \cdot {v_1}^T \cdot I_N +$$

$$v_2 \cdot {M_2}^{-1} \cdot {v_2}^T \cdot I_N + C_1 \cdot {C_1}^T \cdot I_N \eta + C_2 \cdot {C_2}^T \cdot I_N \eta$$

A.4.4 Procedure *Pro_Dual Model*

The last procedure denoted by *Pro_Dual Model*, constructs the dual model representation. The input of this procedure is just the primal model provided by the user. Implementing this procedure for the examples 2 and 3 will result in the following model expressions.

For example 2,

```
> Pro_DualModel(Phi.w+b*N1);
```

$$\frac{1}{2}\,\Phi\Phi^T\cdot\alpha_2+bN1$$

For example 3,

```
> Pro_DualModel([Phi[1].w[1]+b[1]*N1,Phi[2].w[2]+b[2]*N1]);
```

$$\frac{\Phi_1{\Phi_1}^T\cdot(M_1{}^{-1}\cdot{v_1}^T\cdot z-M_1{}^{-1}\cdot b_1N1)}{\gamma}+b_1N1,\ \frac{\Phi_2{\Phi_2}^T\cdot(M_2{}^{-1}\cdot{v_2}^T\cdot z-M_2{}^{-1}\cdot b_2N1)}{\gamma}+b_2N1$$

where

$$M_1=\Phi_1{\Phi_1}^T+\frac{I}{\eta},\ M_2=\Phi_2{\Phi_2}^T+\frac{I}{\eta}.$$

A.5 GUI Application

In order for the code to be user friendly, the Maplet of the code is designed, (see Fig. A.2), containing windows, textbox regions and other visual interfaces, which gives the user point-and-click access. It is an alternative to the worksheet. Users can perform the *SYM-LSSVM-SOLVER* Package without having to get involved in the Maple syntax. Example 1 has been performed using the Maplet package (see Fig. A.2). The snapshots of the result taken from the Maplets are shown in Fig. A.3.

A.6 Conclusion and future work

A symbolic solver written in Maple is developed for LSSVM models. The Maplet of our code is also provided as an alternative to the worksheet. The application of the solver is illustrated on three examples. Currently the LSSVM models that can be handled in our symbolic solver include equality constraints only. Dealing with additional inequality constraints is a further challenge for future work.

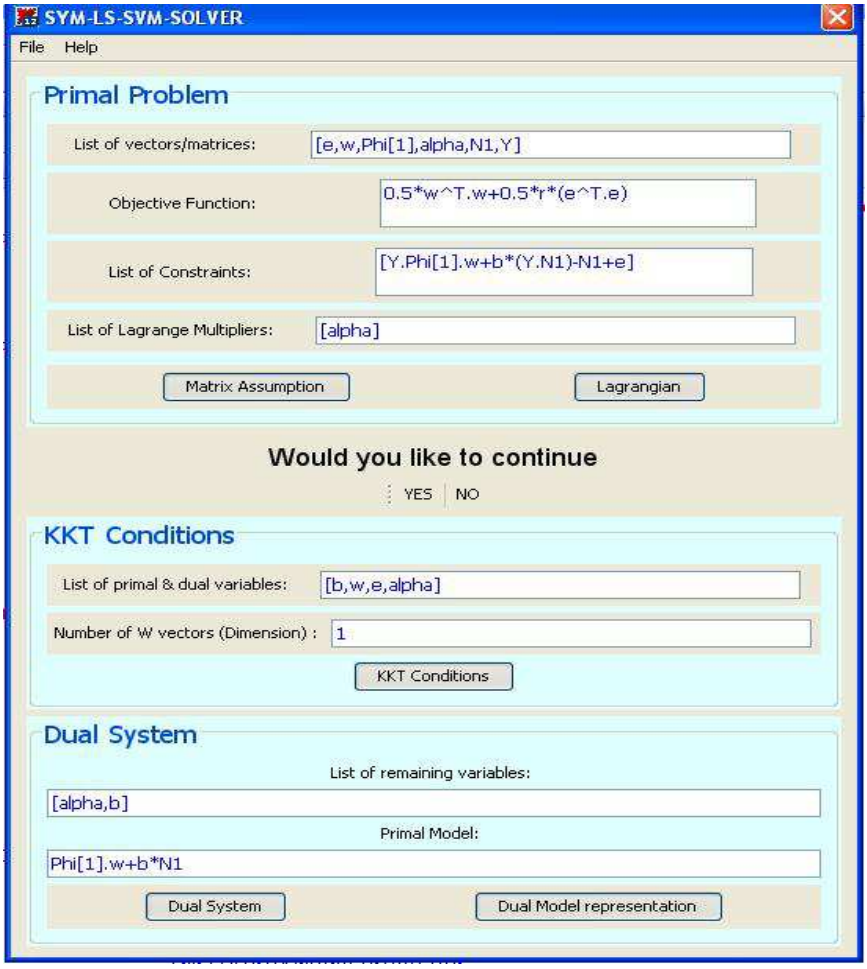
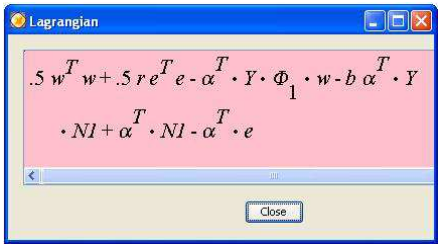
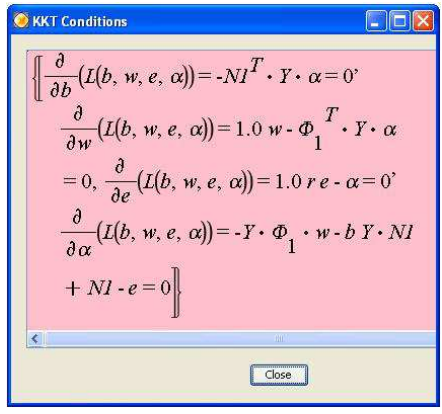


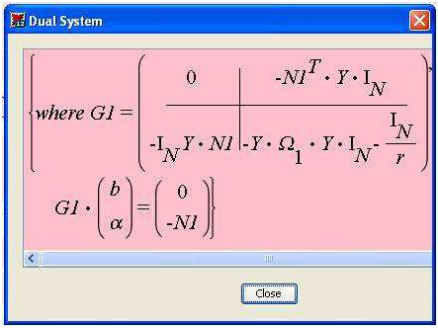
Figure A.2: The GUI for SYM-LSSVM-SOLVER



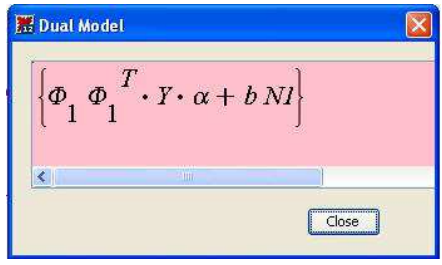
(a)



(b)



(c)



(d)

Figure A.3: Lagrangian function, KKT optimality conditions, Corresponding dual system and model representation for Example 1.

Bibliography

[1] Abushama, A. A., and Bialecki, B. Modified nodal cubic spline collocation for Poisson's equation. *SIAM Journal on Numerical Analysis 46*, 1 (2008), 397–418.

[2] Adankon, M. M., Cheriet, M., and Biem, A. Semi-supervised least squares support vector machine. *IEEE Transactions on Neural Networks 20*, 12 (2009), 1858–1870.

[3] Ahmed, S., Huang, B., and Shah, S. Parameter and delay estimation of continuous-time models using a linear filter. *Journal of Process Control 16*, 4 (2006), 323–331.

[4] Alzate, C., and Suykens, J. A. K. Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 2 (2010), 335–347.

[5] Alzate, C., and Suykens, J. A. K. A semi-supervised formulation to binary kernel spectral clustering. In *The 2012 International Joint Conference on Neural Networks (IJCNN)* (2012), IEEE, pp. 1992–1999.

[6] Ang, W.-T. A numerical method for the wave equation subject to a non-local conservation condition. *Applied numerical mathematics 56*, 8 (2006), 1054–1060.

[7] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 5 (2011), 898–916.

[8] Arun Kumar, M., and Gopal, M. Least squares twin support vector machines for pattern classification. *Expert Systems with Applications 36*, 4 (2009), 7535–7543.

[9] Ascher, U. On symmetric schemes and differential-algebraic equations. *SIAM journal on scientific and statistical computing 10*, 5 (1989), 937–949.

[10] Ascher, U. M., and Petzold, L. R. Projected implicit Runge-Kutta methods for differential-algebraic equations. *SIAM Journal on Numerical Analysis 28*, 4 (1991), 1097–1120.

[11] Ascher, U. M., and Petzold, L. R. *Computer methods for ordinary differential equations and differential-algebraic equations*, vol. 61. Siam, 1998.

[12] Ascher, U. M., and Spiteri, R. J. Collocation software for boundary value differential-algebraic equations. *SIAM Journal on Scientific Computing 15*, 4 (1994), 938–952.

[13] Asuncion, A., and Newman, D. J. UCI machine learning repository, 2007.

[14] Awawdeh, F., Jaradat, H., and Alsayyed, O. Solving system of daes by homotopy analysis method. *Chaos, Solitons & Fractals 42*, 3 (2009), 1422–1427.

[15] Badrinarayanan, V., Budvytis, I., and Cipolla, R. Semi-supervised video segmentation using tree structured graphical models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 35*, 11 (2013), 2751–2764.

[16] Baker, C. T., and Baker, C. *The numerical treatment of integral equations*, vol. 13. Clarendon press Oxford, 1977.

[17] Banks, H., Burns, J., and Cliff, E. Parameter estimation and identification for systems with delays. *SIAM Journal on Control and Optimization 19*, 6 (1981), 791–828.

[18] Banks, H. T., and Lamm, P. D. Estimation of delays and other parameters in nonlinear functional differential equations. *SIAM journal on control and optimization 21*, 6 (1983), 895–915.

[19] Barrero, O. *Data assimilation in magnetohydrodynamics systems using Kalman Filtering*. PhD thesis, Katholieke Universiteit Leuven (KU Leuven), Leuven (Belgium), November 2005.

[20] Batzel, J. J., and Tran, H. Stability of the human respiratory control system i. analysis of a two-dimensional delay state-space model. *Journal of mathematical biology 41*, 1 (2000), 45–79.

[21] Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research 7* (2006), 2399–2434.

[22] Bellen, A., and Zennaro, M. *Numerical methods for delay differential equations*. Oxford University Press, 2013.

[23] Bellman, R., Jacquez, J., Kalaba, R., and Schwimmer, S. Quasilinearization and the estimation of chemical rate constants from raw kinetic data. *Mathematical Biosciences 1*, 1 (1967), 71–76.

[24] Bezdek, J. C., and Pal, N. R. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 28*, 3 (1998), 301–315.

[25] Biegler, L., Damiano, J., and Blau, G. Nonlinear parameter estimation: a case study comparison. *AIChE Journal 32*, 1 (1986), 29–45.

[26] Biegler, L., Damiano, J., and Blau, G. Nonlinear parameter estimation: a case study comparison. *AIChE Journal 32*, 1 (1986), 29–45.

[27] Bishop, C. M., and Nasrabadi, N. M. *Pattern recognition and machine learning*, vol. 1. Springer New York, 2006.

[28] Bonilla, J., Diehl, M., De Moor, B., and Van Impe, J. A nonlinear least squares estimation procedure without initial parameter guesses. In *2008 47th IEEE Conference on Decision and Control* (2008), pp. 5519–5524.

[29] Borsotti, M., Campadelli, P., and Schettini, R. Quantitative evaluation of color image segmentation results. *Pattern recognition letters 19*, 8 (1998), 741–747.

[30] Boyd, J. P. Six strategies for defeating the Runge phenomenon in gaussian radial basis functions on a finite interval. *Computers & Mathematics with Applications 60*, 12 (2010), 3108–3122.

[31] Boyd, J. P., and Wang, L. Asymptotic coefficients for gaussian radial basis function interpolants. *Applied Mathematics and Computation 216*, 8 (2010), 2394–2407.

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

[33] Brenan, K. Numerical simulation of trajectory prescribed path control problems by the backward differentiation formulas. *Automatic Control, IEEE Transactions on 31*, 3 (1986), 266–269.

[34] Brenan, K. E., Campbell, S. L., and Petzold, L. R. *Numerical solution of initial-value problems in differential-algebraic equations*, vol. 14. Siam, 1996.

[35] Butcher, J. C. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008.

[36] Campbell, S. L. V., and Campbell, S. *Singular systems of differential equations*, vol. 1. Pitman London, 1980.

[37] Chakrabarti, D., Kumar, R., and Tomkins, A. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), ACM, pp. 554–560.

[38] Chakraborty, S., and Nagwani, N. Analysis and study of incremental k-means clustering algorithm. In *High Performance Architecture and Grid Computing*. Springer, 2011, pp. 338–341.

[39] Chang, C. C., and Lin, C. J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST) 2*, 3 (2011), 27:1–27:27.

[40] Chang, C. C., Pao, H. K., and Lee, Y. J. An rsvm based two-teachers–one-student semi-supervised learning algorithm. *Neural Networks 25* (2012), 57–69.

[41] Chapelle, O., Schölkopf, B., and Zien, A. *Semi-supervised learning*, vol. 2. MIT press Cambridge, 2006, Eds.

[42] Chapelle, O., Sindhwani, V., and Keerthi, S. Branch and bound for semi-supervised support vector machines. *NIPS* (2006), 217–224.

[43] Chawla, M., and Katti, C. Finite difference methods for two-point boundary value problems involving high order differential equations. *BIT Numerical Mathematics 19*, 1 (1979), 27–33.

[44] Chen, S. Kalman filter for robot vision: a survey. *Industrial Electronics, IEEE Transactions on 59*, 11 (2012), 4409–4420.

[45] Cheng, J., Sayeh, M. R., Zargham, M. R., and Cheng, Q. Real-time vector quantization and clustering based on ordinary differential equations. *Neural Networks, IEEE Transactions on 22*, 12 (2011), 2143–2148.

[46] Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. L. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (2007), ACM, pp. 153–162.

[47] Christodoulou, C., and Georgiopoulos, M. *Applications of neural networks in electromagnetics*. Artech House, Inc., 2000.

[48] Chung, F. R. *Spectral graph theory*, vol. 92. AMS Bookstore, 1997.

[49] Clark, K. D., and Petzold, L. R. Numerical solution of boundary value problems in differential-algebraic systems. *SIAM journal on scientific and statistical computing 10*, 5 (1989), 915–936.

[50] Cristianini, N., and Shawe-Taylor, J. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[51] De Brabanter, K., De Brabanter, J., Suykens, J. A. K., and De Moor, B. Optimized fixed-size kernel models for large data sets. *Computational Statistics & Data Analysis 54*, 6 (2010), 1484–1504.

[52] De Brabanter, K., De Brabanter, J., Suykens, J. A. K., and De Moor, B. Approximate confidence and prediction intervals for least squares support vector regression. *Neural Networks, IEEE Transactions on 22*, 1 (2011), 110–120.

[53] Dehghan, M., and Taleei, A. A compact split-step finite difference method for solving the nonlinear schrödinger equations with constant and variable coefficients. *Computer Physics Communications 181*, 1 (2010), 43–51.

[54] Dua, V. An artificial neural network approximation based decomposition approach for parameter estimation of system of ordinary differential equations. *Computers & chemical engineering 35*, 3 (2011), 545–553.

[55] Edsberg, L., and Wikström, G. Toolbox for parameter estimation and simulation in: dynamic systems with applications to chemical kinetics. In *Proceedings of the Nordic MATLAB Conference* (1995).

[56] Ellner, S. P., Kendall, B. E., Wood, S. N., McCauley, E., and Briggs, C. J. Inferring mechanism from time-series data: delay-differential equations. *Physica D: Nonlinear Phenomena 110*, 3 (1997), 182–194.

[57] Espinoza, M., Suykens, J. A. K., and De Moor, B. Fixed-size least squares support vector machines: A large scale application in electrical load forecasting. *Computational Management Science 3*, 2 (2006), 113–129.

[58] Fortunato, S. Community detection in graphs. *Physics Reports 486*, 3 (2010), 75–174.

[59] Franklin, G., Powell, J., and Workman, M. *Digital Control of Dynamic Systems*, second ed. Addison-Wesley, 1990.

[60] Froment, G. F., Bischoff, K. B., De Wilde, J., et al. *Chemical reactor analysis and design*, vol. 2. Wiley New York, 1990.

[61] Fung, G., and Mangasarian, O. L. Proximal support vector machine classifiers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (2001), ACM, pp. 77–86.

[62] Gauthier, B., and Pronzato, L. Spectral approximation of the imse criterion for optimal designs in kernel-based interpolation models. *SIAM/ASA Journal on Uncertainty Quantification 2*, 1 (2014), 805–825.

[63] Gear, C. W., and Petzold, L. R. ODE methods for the solution of differential/algebraic systems. *SIAM Journal on Numerical Analysis 21*, 4 (1984), 716–728.

[64] Girolami, M. Orthogonal series density estimation and the kernel eigenvalue problem. *Neural Computation 14*, 3 (2002), 669–688.

[65] Girvan, M., and Newman, M. E. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences 99*, 12 (2002), 7821–7826.

[66] Golub, G. H., and Van Loan, C. F. *Matrix computations*. Johns Hopkins University Press, 2012.

[67] Grundmann, M., Kwatra, V., Han, M., and Essa, I. Efficient hierarchical graph-based video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition* (2010), IEEE, pp. 2141–2148.

[68] Guzel, N., and Bayram, M. Numerical solution of differential–algebraic equations with index-2. *Applied Mathematics and Computation 174*, 2 (2006), 1279–1289.

[69] Hackbusch, W. *Multi-grid methods and applications*, vol. 4. Springer-Verlag Berlin, 1985.

[70] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. On clustering validation techniques. *Journal of Intelligent Information Systems 17*, 2-3 (2001), 107–145.

[71] Hartung, F. Parameter estimation by quasilinearization in functional differential equations with state-dependent delays: a numerical study. *Nonlinear Analysis: Theory, Methods & Applications 47*, 7 (2001), 4557–4566.

[72] Hartung, F., and Turi, J. Identification of parameters in neutral functional differential equations with state-dependent delays. In *IEEE conference on decision and control* (2005), vol. 44, 1998, p. 5239.

[73] Heckbert, P. *Color image quantization for frame buffer display*, vol. 16. ACM, 1982.

[74] Horton, G., and Vandewalle, S. A space-time multigrid method for parabolic partial differential equations. *SIAM Journal on Scientific Computing 16*, 4 (1995), 848–864.

[75] Huang, C., and Vandewalle, S. Unconditionally stable difference methods for delay partial differential equations. *Numerische Mathematik 122*, 3 (2012), 579–601.

[76] Huang, X., Shi, L., and Suykens, J. A. K. Support vector machine classifier with pinball loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[77] Huang, Y., Xu, D., and Nie, F. Semi-supervised dimension reduction using trace ratio criterion. *Neural Networks and Learning Systems, IEEE Transactions on 23*, 3 (2012), 519–526.

[78] Jerri, A. J. *Advances in The Gibbs Phenomenon with Detailed Introduction*. Clarkson University, 2007.

[79] Jia, J., and Wang, H. Fast finite difference methods for space-fractional diffusion equations with fractional derivative boundary conditions. *Journal of Computational Physics* (2014).

[80] Kalman, R. E. A new approach to linear filtering and prediction problems. *Transactions ASME, Series D, Journal of basic engineering 82* (March 1960), 34–45.

[81] Kalmár-Nagy, T., Stépán, G., and Moon, F. C. Subcritical hopf bifurcation in the delay equation model for machine tool vibrations. *Nonlinear Dynamics 26*, 2 (2001), 121–142.

[82] Karasuyama, M., and Mamitsuka, H. Multiple graph label propagation by sparse integration. *IEEE Transactions on Neural Networks and Learning Systems 24*, 12 (2013), 1999–2012.

[83] Khemchandani, R., Chandra, S., et al. Twin support vector machines for pattern classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 29*, 5 (2007), 905–910.

[84] Koenker, R. *Quantile regression*. No. 38. Cambridge university press, 2005.

[85] Kolmanovskii, V., and Myshkis, A. *Applied theory of functional differential equations*. Springer, 1992.

[86] Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *Neural Networks, IEEE Transactions on 9*, 5 (1998), 987–1000.

[87] Lagaris, I. E., Likas, A. C., and Papageorgiou, D. G. Neural-network methods for boundary value problems with irregular boundaries. *Neural Networks, IEEE Transactions on 11*, 5 (2000), 1041–1049.

[88] Lai, C. Y., Xiang, C., and Lee, T. H. Data-based identification and control of nonlinear systems via piecewise affine approximation. *Neural Networks, IEEE Transactions on 22*, 12 (2011), 2189–2200.

[89] Lakestani, M., and Dehghan, M. Collocation and finite difference-collocation methods for the solution of nonlinear klein–gordon equation. *Computer Physics Communications 181*, 8 (2010), 1392–1401.

[90] Lambert, J. D. *Numerical methods for ordinary differential systems: the initial value problem*. John Wiley & Sons, Inc., 1991.

[91] Lázaro, M., Santamaría, I., Pérez-Cruz, F., and Artés-Rodríguez, A. Support vector regression for the simultaneous learning of a multivariate function and its derivatives. *Neurocomputing 69*, 1 (2005), 42–61.

[92] Lázaro, M., Santamaría, I., Pérez-Cruz, F., and Artés-Rodríguez, A. Support vector regression for the simultaneous learning of a multivariate function and its derivatives. *Neurocomputing 69*, 1 (2005), 42–61.

[93] Lee, H., and Kang, I. S. Neural algorithm for solving differential equations. *Journal of Computational Physics 91*, 1 (1990), 110–131.

[94] Lee, Y.-J., and Mangasarian, O. L. RSVM: Reduced support vector machines. In *Proceedings of the first SIAM international conference on data mining* (2001), SIAM Philadelphia, pp. 5–7.

[95] LeVeque, R. J. *Finite volume methods for hyperbolic problems*, vol. 31. Cambridge university press, 2002.

[96] Li, P., Li, Y., and Seem, J. E. Consistent initialization of system of differential-algebraic equations for dynamic simulation of centrifugal chillers. *Journal of Building Performance Simulation 5*, 2 (2012), 115–139.

[97] Li, Y.-F., Kwok, J. T., and Zhou, Z.-H. Semi-supervised learning using label mean. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 633–640.

[98] Ljung, L. *System identification*. Springer, 1998.

[99] Lorenz, E. N. Deterministic nonperiodic flow. *Journal of the atmospheric sciences 20*, 2 (1963), 130–141.

[100] Lötstedt, P., and Petzold, L. Numerical solution of nonlinear differential equations with algebraic constraints. i. convergence results for backward differentiation formulas. *Mathematics of computation 46*, 174 (1986), 491–516.

[101] Ma, X., Gao, L., Yong, X., and Fu, L. Semi-supervised clustering algorithm for community structure detection in complex networks. *Physica A: Statistical Mechanics and its Applications 389*, 1 (2010), 187–197.

[102] Macskassy, S. A., and Provost, F. Classification in networked data: A toolkit and a univariate case study. *The Journal of Machine Learning Research 8* (2007), 935–983.

[103] Mangasarian, O. L., and Wild, E. W. Multisurface proximal support vector machine classification via generalized eigenvalues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 1 (2006), 69–74.

[104] Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *in Proc. 8th International Conference on Computer Vision* (2001), vol. 2, IEEE, pp. 416–423.

[105] Maz'ya, V., and Schmidt, G. On approximate approximations using gaussian kernels. *IMA Journal of Numerical Analysis 16*, 1 (1996), 13–29.

[106] McFall, K. S., and Mahan, J. R. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *IEEE Transactions on Neural Networks 20*, 8 (2009), 1221–1233.

[107] Meade Jr, A. J., and Fernandez, A. A. The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling 19*, 12 (1994), 1–25.

[108] Mehrkanoon, S., Agudelo, M., and Suykens, J. A. K. Incremental multi-class semi-supervised clustering regularized by kalman filtering. *Internal Report 14-154, ESAT-SISTA, KU Leuven (Leuven, Belgium)* (2014,Lirias number:x).

[109] Mehrkanoon, S., Alzate, C., Mall, R., Langone, R., and Suykens, J. A. K. Multiclass semi-supervised learning based upon kernel spectral clustering. *IEEE Transactions on Neural Networks and Learning Systems 26*, 4 (2015), 720–733.

[110] Mehrkanoon, S., Falck, T., and Suykens, J. A. K. Approximate solutions to ordinary differential equations using least squares support vector machines. *IEEE Transactions on Neural Networks and Learning Systems 23*, 9 (2012), 1356–1367.

[111] Mehrkanoon, S., Falck, T., and Suykens, J. A. K. Parameter estimation for time varying dynamical systems using least squares support vector machines. In *Proc. of the 16th IFAC Symposium on System Identification (SYSID 2012), Brussels, Belgium* (2012), pp. 1300–1305.

[112] Mehrkanoon, S., Huang, X., and Suykens, J. A. K. Non-parallel support vector classifiers with different loss functions. *Neurocomputing 143* (2014), 294–301.

[113] Mehrkanoon, S., Jiang, L., Alzate, C., and Suykens, J. A. K. Symbolic computing of ls-svm based models. In *In proc of the 19th European Symposium on Artificial Neural Networks (ESANN)* (2011), pp. 183–188.

[114] Mehrkanoon, S., Mehrkanoon, S., and Suykens, J. A. K. Parameter estimation of delay differential equations: an integration-free ls-svm approach. *Communications in Nonlinear Science and Numerical Simulation 19*, 4 (2014), 830–841.

[115] Mehrkanoon, S., and Suykens, J. A. K. Non-parallel semi-supervised classification based on kernel spectral clustering. In *The 2013 International Joint Conference on Neural Networks (IJCNN)* (2012), IEEE, pp. 2311–2318.

[116] Mehrkanoon, S., and Suykens, J. A. K. LS-SVM approximate solution to linear time varying descriptor systems. *Automatica 48*, 10 (2012), 2502–2511.

[117] Mehrkanoon, S., and Suykens, J. A. K. Large scale semi-supervised learning using ksc based model. In *Neural Networks (IJCNN), 2014 International Joint Conference on* (2014), IEEE, pp. 4152–4159.

[118] Mehrkanoon, S., and Suykens, J. A. K. Learning solutions to partial differential equations using ls-svm. *Neurocomputing 159* (2015), 105–116.

[119] Melacci, S., and Belkin, M. Laplacian support vector machines trained in the primal. *The Journal of Machine Learning Research 12* (2011), 1149–1184.

[120] Mohanty, R. An unconditionally stable finite difference formula for a linear second order one space dimensional hyperbolic equation with variable coefficients. *Applied Mathematics and Computation 165*, 1 (2005), 229–236.

[121] Moles, C. G., Mendes, P., and Banga, J. R. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome research 13*, 11 (2003), 2467–2474.

[122] Murugesan, K., and Balakumar, V. Study on singular systems of index three via runge-kutta method of order-10. *International Journal of Pure and Applied Mathematics 70*, 5 (2011), 723–733.

[123] Murugesan, K., Dhayabaran, D. P., and Amirtharaj, E. H. A study of second-order state-space systems of time-invariant and time-varying transistor circuits using the stws technique. *International journal of electronics 89*, 4 (2002), 305–315.

[124] Murugesh, V., and Murugesan, K. RK–Butcher algorithms for singular system-based electronic circuit. *International Journal of Computer Mathematics 86*, 3 (2009), 523–536.

[125] Myshkis, A. D. Razumikhin's method in the qualitative theory of processes with delay. *International Journal of Stochastic Analysis 8*, 3 (1995), 233–247.

[126] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The Computer Journal 7*, 4 (1965), 308–313.

[127] Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems 2* (2002), 849–856.

[128] Nie, F., Zeng, Z., Tsang, I. W., Xu, D., and Zhang, C. Spectral embedded clustering: a framework for in-sample and out-of-sample spectral clustering. *Neural Networks, IEEE Transactions on 22*, 11 (2011), 1796–1808.

[129] Nikolaev, N. Y., and Iba, H. Learning polynomial feedforward neural networks by genetic programming and backpropagation. *IEEE Transactions on Neural Networks 14*, 2 (2003), 337–350.

[130] Ning, H., Jing, X., and Cheng, L. Online identification of nonlinear spatiotemporal systems using kernel learning approach. *Neural Networks, IEEE Transactions on 22*, 9 (2011), 1381–1394.

[131] Ning, H., Xu, W., Chi, Y., Gong, Y., and Huang, T. S. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition 43*, 1 (2010), 113–127.

[132] Peng, X. Efficient twin parametric insensitive support vector regression model. *Neurocomputing 79* (2012), 26–38.

[133] Peng, X., and Xu, D. Bi-density twin support vector machines for pattern recognition. *Neurocomputing 99* (2013), 134–143.

[134] Petersen, K. B., and Pedersen, M. S. The matrix cookbook, 2006.

[135] Quirynen, R., Gros, S., and Diehl, M. Fast auto generated acado integrators and application to mhe with multi-rate measurements. In *Control Conference (ECC), 2013 European* (2013), IEEE, pp. 3077–3082.

[136] Rabier, P. J., and Rheinboldt, W. C. Time-dependent linear daes with discontinuous inputs. *Linear algebra and its applications 247* (1996), 1–29.

[137] Ramuhalli, P., Udpa, L., and Udpa, S. S. Finite-element neural networks for solving differential equations. *Neural Networks, IEEE Transactions on 16*, 6 (2005), 1381–1392.

[138] Ranade, V. V. *Computational flow modeling for chemical reactor engineering*, vol. 5. Academic press, 2001.

[139] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[140] Russell, R. D., and Shampine, L. A collocation method for boundary value problems. *Numerische Mathematik 19*, 1 (1972), 1–28.

[141] Sarlin, P. Self-organizing time map: an abstraction of temporal multivariate patterns. *Neurocomputing 99* (2013), 496–508.

[142] Schölkopf, B., and Smola, A. J. *Learning with kernels*. MIT Press, 2002.

[143] Shampine, L. F., and Thompson, S. Solving ddes in matlab. *Applied Numerical Mathematics 37*, 4 (2001), 441–458.

[144] Shao, Y.-H., Deng, N.-Y., and Yang, Z.-M. Least squares recursive projection twin support vector machine for classification. *Pattern Recognition 45*, 6 (2012), 2299–2307.

[145] Shao, Y.-H., Zhang, C.-H., Wang, X.-B., and Deng, N.-Y. Improvements on twin support vector machines. *Neural Networks, IEEE Transactions on 22*, 6 (2011), 962–968.

[146] Shekari Beidokhti, R., and Malek, A. Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques. *Journal of the Franklin Institute 346*, 9 (2009), 898–913.

[147] Shiang, K.-D. A perturbation-based estimate algorithm for parameters of coupled ordinary differential equations, applications from chemical reactions to metabolic dynamics. *Computer methods and programs in biomedicine 94*, 2 (2009), 118–142.

[148] Sindhwani, V., and Keerthi, S. S. Large scale semi-supervised linear SVMs. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), ACM, pp. 477–484.

[149] Sóbester, A., Nair, P. B., and Keane, A. J. Genetic programming approaches for solving elliptic partial differential equations. *IEEE Transactions on Evolutionary Computation 12*, 4 (2008), 469–478.

[150] Steinwart, I., Christmann, A., et al. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli 17*, 1 (2011), 211–225.

[151] Strikwerda, J. C. *Finite difference schemes and partial differential equations*. Siam, 2004.

[152] Suliman, C., Cruceru, C., and Moldoveanu, F. Kalman filter based tracking in an video surveillance system. *Advances in Electrical and Computer Engineering 10*, 2 (2010), 30–34.

[153] Sun, Y., Wong, A. K., and Kamel, M. S. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence 23*, 04 (2009), 687–719.

[154] Suykens, J. A. K. Data visualization and dimensionality reduction using kernel maps with a reference point. *IEEE Transactions on Neural Networks 19*, 9 (2008), 1501–1517.

[155] Suykens, J. A. K., Alzate, C., and Pelckmans, K. Primal and dual model representations in kernel-based learning. *Statistics Surveys 4* (2010), 148–183.

[156] Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., and Vandewalle, J. *Least squares support vector machines*. World Scientific, 2002.

[157] Suykens, J. A. K., and Vandewalle, J. Least squares support vector machine classifiers. *Neural processing letters 9*, 3 (1999), 293–300.

[158] Suykens, J. A. K., Vandewalle, J., and De Moor, B. Optimal control by least squares support vector machines. *Neural Networks 14*, 1 (2001), 23–35.

[159] Tan, K. S., Mat Isa, N. A., and Lim, W. H. Color image segmentation using adaptive unsupervised clustering approach. *Applied Soft Computing 13*, 4 (2013), 2017–2036.

[160] Teichman, A., and Thrun, S. Tracking-based semi-supervised learning. *The International Journal of Robotics Research 31*, 7 (2012), 804–818.

[161] Thomée, V. From finite differences to finite elements: A short history of numerical analysis of partial differential equations. *Journal of Computational and Applied Mathematics 128*, 1 (2001), 1–54.

[162] Toselli, A., and Widlund, O. B. *Domain decomposition methods: algorithms and theory*, vol. 34. Springer, 2005.

[163] Tsoulos, I. G., Gavrilis, D., and Glavas, E. Solving differential equations with constructed neural networks. *Neurocomputing 72*, 10 (2009), 2385–2391.

[164] Tsoulos, I. G., and Lagaris, I. E. Solving differential equations with genetic programming. *Genetic Programming and Evolvable Machines 7*, 1 (2006), 33–54.

[165] van Milligen, B. P., Tribaldos, V., and Jiménez, J. Neural network differential equation and plasma equilibrium solver. *Physical review letters 75*, 20 (1995), 3594.

[166] Vandewalle, S., and Piessens, R. Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations. *SIAM journal on scientific and statistical computing 13*, 6 (1992), 1330–1346.

[167] Vapnik, V. *Statistical learning theory*. Wiley, 1998.

[168] Varah, J. A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific and Statistical Computing 3*, 1 (1982), 28–46.

[169] Versteeg, H. K., and Malalasekera, W. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[170] Vinh, N. X., Epps, J., and Bailey, J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 1073–1080.

[171] von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing 17*, 4 (2007), 395–416.

[172] Wang, H.-S., Jhu, W.-L., Yung, C.-F., and Wang, P.-F. Numerical solutions of differential-algebraic equations and its applications in solving tppc problems. *Journal of Marine Science and Technology 19*, 1 (2011), 76–88.

[173] Wang, L., and Cao, J. Estimating parameters in delay differential equation models. *Journal of agricultural, biological, and environmental statistics 17*, 1 (2012), 68–83.

[174] Wang, Y., Chen, S., and Zhou, Z.-H. New semi-supervised classification method based on modified cluster assumption. *IEEE Transactions on Neural Networks and Learning Systems 23*, 5 (2012), 689–702.

[175] Warren Liao, T. Clustering of time series dataŮa survey. *Pattern recognition 38*, 11 (2005), 1857–1874.

[176] Weng, S.-K., Kuo, C.-M., and Tu, S.-K. Video object tracking using adaptive kalman filter. *Journal of Visual Communication and Image Representation 17*, 6 (2006), 1190–1208.

[177] Williams, C., and Seeger, M. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13* (2001).

[178] Wood, S. N. Partially specified ecological models. *Ecological Monographs 71*, 1 (2001), 1–25.

[179] Xavier-De-Souza, S., Suykens, J. A. K., Vandewalle, J., and Bollé, D. Coupled simulated annealing. *IEEE Trans. Sys. Man Cyber. Part B 40*, 2 (Apr. 2010), 320–335.

[180] Xiang, S., Nie, F., and Zhang, C. Semi-supervised classification via local spline regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 11 (2010), 2039–2053.

[181] Yang, Y., Nie, F., Xu, D., Luo, J., Zhuang, Y., and Pan, Y. A multimedia retrieval framework based on semi-supervised ranking and relevance feedback. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 4 (2012), 723–742.

[182] Zachary, W. W. An information flow model for conflict and fission in small groups. *Journal of anthropological research* (1977), 452–473.

[183] Zhang, Z. Community structure detection in complex networks with partial background information. *Europhysics Letters 101*, 48005 (2013).

[184] Zhao, Y.-P., Zhao, J., and Zhao, M. Twin least squares support vector regression. *Neurocomputing 118* (2013), 225–236.

[185] Zhong, J., and Sclaroff, S. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (2003), IEEE, pp. 44–50.

[186] Zhu, X. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison* (2006).

List of publications

International Journal Publications

S. Mehrkanoon Yuri A.W. Shardt, J.A.K. Suykens and Steven X. Ding, Estimating the Unknown Time Delay in Chemical Processes, Internal Report 14-154, ESAT-SISTA, KU Leuven (Leuven, Belgium), 2015, submitted.

S. Mehrkanoon M. Agudelo, J.A.K. Suykens, Incremental multi-class semi-supervised clustering regularized by Kalman filtering. Internal Report 14-154, ESAT-SISTA, KU Leuven (Leuven, Belgium), 2014, submitted.

S. Mehrkanoon and J. A. K. Suykens, Learning Solutions to Partial Differential Equations using LS-SVM. *Neurocomputing*, vol. 159, Mar. 2015, pp. 105-116.

Raghvendra Mall, **S. Mehrkanoon** and J. A. K. Suykens, Identifying Intervals for Hierarchical Clustering using the Gershgorin Circle Theorem. *Pattern Recognition Letters*, vol 55, pp. 1-7, 2015.

S. Mehrkanoon, C. Alzate, R. Mall, R. Langone, J. A. K. Suykens, Multi-class semi-supervised learning based upon kernel spectral clustering. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, Apr. 2015, pp. 720-733.

S. Mehrkanoon, X. Huang, J. A. K. Suykens, Non-parallel Classifiers with Different Loss Functions. *Neurocomputing*, vol. 143, Nov. 2014, pp. 294-301, 2014.

S. Mehrkanoon, S. Mehrkanoon, J. A. K. Suykens, Parameter estimation of delay differential equations: an integration-free LS-SVM approach. *Communications in Nonlinear Science and Numerical Simulation*, Vol. 19, 830-841, 2014.

X. Huang, **S. Mehrkanoon**, J. A. K. Suykens, Support vector machines with piecewise linear feature mapping. *Neurocomputing*, Vol. 117, 118-127, 2013.

S. Mehrkanoon, J. A. K. Suykens, LS-SVM approximate solution to linear time varying descriptor systems. *Automatica*, 48(10), 2502-2511, 2012.

S. Mehrkanoon, T. Falck, J. A. K. Suykens, Approximate solutions to ordinary differential equations using least squares support vector machines. *IEEE Transactions on Neural Networks and Learning Systems*, 23(9), 1356-1367, 2012.

Conference proceedings

S. Mehrkanoon, M. Agudelo, R.Mall and J.A.K. Suykens, Hierarchical Semi-Supervised Clustering using KSC based model, *Accepted for publication in Proc. of the International Joint Conference on Neural Networks 2015.*

Z. Karevan, **S. Mehrkanoon** and J.A.K. Suykens, Black-box modeling for temperature prediction in weather forecasting, *Accepted for publication in Proc. of the International Joint Conference on Neural Networks 2015.*

S. Mehrkanoon, L. Jiang, C. Alzate, J. A. K. Suykens, Symbolic computing of LS-SVM based models. *19th European Symposium on Artificial Neural Networks (ESANN 2011),* Bruges, Belgium, Apr. 2011, pp. 183-188.

S. Mehrkanoon, T. Falck, J. A. K. Suykens, Parameter Estimation for Time Varying Dynamical Systems using Least Squares Support Vector Machines. *16th IFAC Symposium on System Identification (SYSID 2012)* Brussels, Belgium, Jul. 2012, pp. 1300-1305.

S. Mehrkanoon, J. A. K. Suykens, LS-SVM based solution for delay differential equations. *International Conference on Mathematical Modelling in Physical Sciences (IC-MSQUARE 2012)* Budapest, Hungary , Sep. 2012.

S. Mehrkanoon, J. A. K. Suykens, Non-parallel semi-supervised classification based on kernel spectral clustering. *in Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Dallas, U.S.A, Aug. 2013, pp. 2311-2318.

S. Mehrkanoon, R. Quirynen, M. Diehl and J. A. K. Suykens, LSSVM based initialization approach for parameter estimation of dynamical systems.

International Conference on Mathematical Modelling in Physical Sciences (IC-MSQUARE 2013), Prague, Czech Republic, Sep. 2013.

S. Mehrkanoon, J. A. K. Suykens, Large scale semi-supervised learning using KSC based model. *in Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, Jul. 2014, pp. 4152-4159.

R. Castro, **S. Mehrkanoon**, A. Marconato, J. Schoukens and J. A. K. Suykens, SVD truncation schemes for fixed-size kernel models. *in Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, Jun. 2014, pp. 3922-3929.

R. Mall, **S. Mehrkanoon**, R. Langone, J.A.K. Suykens, Optimal Reduced Sets for Sparse Kernel Spectral Clustering, *in Proc. of the International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, Jun. 2014, pp. 2436-2443.

Abstracts

S. Mehrkanoon, J. A. K. Suykens. LS-SVM approach for solving linear descriptor, *31st Benelux Meeting on Systems and Control Heijderbos, Heijen/Nijmegen, The Netherlands*, March 27-29, 2012.

S. Mehrkanoon, J. A. K. Suykens. LS-SVM approximate solutions to PDEs, *33st Benelux Meeting on Systems and Control Heijderbos, Heijen/Nijmegen, The Netherlands*, March 25-27, 2014.

S. Mehrkanoon, J. A. K. Suykens, Non-parallel semi-supervised classification. *International workshop on advances in Regularization, Optimization, Kernel Methods and Support Vector Machines: theory and applications (ROKS 2013)* July 8 - 10, 2013, Leuven, Belgium.

S. Mehrkanoon, J. A. K. Suykens, Time Varying Parameter Estimation in a System of ODEs Using Least Squares Support Vector Machines. *OPTEC Workshop on Moving Horizon Estimation and System Identification MHE 2012*, 29-30 August, 2012, Leuven, Belgium.

S. Mehrkanoon, J. A. K. Suykens, LS-SVM approximate solution to linear time varying descriptor systems. In *DYSCO Study Day.* Oct 8, 2012.

S. Mehrkanoon, J. A. K. Suykens, Parameter estimation of delay differential equations: an integration-free LS-SVM approach. In *DYSCO Study Day.* May 24, 2013.

S. Mehrkanoon, J. A. K. Suykens, LS-SVM approximate solutions to PDEs. In *DYSCO Study Day.* November 22, 2013.

S. Mehrkanoon, J. A. K. Suykens, Online video segmentation using semi-supervised learning. In *DYSCO Study Day.* November 12, 2014.

S. Mehrkanoon, J. A. K. Suykens, Online video segmentation using semi-supervised learning. *European Research Network on System Identification (ERNSI 2014).* September 21-24, Ostend, Belgium.



FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING (ESAT)
STADIUS CENTER FOR DYNAMICAL SYSTEMS, SIGNAL PROCESSING AND DATA ANALYTICS
Kasteelpark Arenberg 10 bus 2446
B-3001 Heverlee
siamak.mehrkanoon@esat.kuleuven.be
https://sites.google.com/site/smkmhr/home