THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Machine Learning Methods Using Class-specific Subspace Kernel Representations

for Large-Scale Applications

Yinan Yu

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2016

Machine Learning Methods Using Class-specific Subspace Kernel Representations

for Large-Scale Applications

Yinan Yu

ISBN 978-91-7597-487-3

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 4168
ISSN 0346-718X

Department of Signals and Systems
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone + 46 (0)31 – 772 1000

Typeset by the author using LaTeX.

Chalmers Reproservice
Göteborg, Sweden 2016

To my family

Abstract

Kernel techniques became popular due to and along with the rising success of Support Vector Machines (SVM). During the last two decades, the kernel idea itself has been extracted from SVM and is now widely studied as an independent subject. Essentially, kernel methods are nonlinear transformation techniques that take data from an input set to a high (possibly infinite) dimensional vector space, called the Reproducing Kernel Hilbert Space (RKHS), in which linear models can be applied. The original input set could be data from different domains and applications, such as tweets, ratings of movies, images, medical measurements, etc. The two spaces are connected by a Positive-Semi Definite (PSD) *kernel function* and all computations in the RKHS are evaluated on the low dimensional input set using the kernel function.

Kernel methods are proven to be efficient on various applications. However, the computational complexity of most kernel algorithms typically grows cubically, or at least quadratically, with respect to the training size. This is due to the fact that a Gram kernel matrix needs to be constructed and/or inverted. To improve the scalability for large-scale training, kernel approximation techniques are employed, where the kernel matrix is assumed to have a low-rank structure. Essentially, this is equivalent to assuming a subspace model spanned by a subset of the training data in the RKHS. The task is hence to estimate the subspace with respect to some criteria, such as the reconstruction error, the discriminative power for classification tasks, etc.

Based on these motivations, this thesis focuses on the development of scalable kernel techniques for supervised classification problems. Inspired by the idea of the subspace classifier and kernel clustering models, we have proposed the CLAss-specific Subspace Kernel (CLASK) representation, where class-specific kernel functions are applied and individual subspaces can be constructed accordingly. In this thesis work, an automatic model selection technique is proposed to choose the best multiple kernel functions for each class based on a criterion using the subspace projection distance. Moreover, subset selection and transformation techniques using CLASK are developed to further reduce the model complexity with an enhanced discriminative power for kernel approximation and classification. Furthermore, we have also proposed both a parallel and a sequential framework to tackle large-scale learning problems.

Acknowledgement

I would like to take this opportunity to thank Prof. Mats Viberg and Prof. Tomas McKelvey for giving me the opportunity to join the Signal Processing group as a PhD candidate. I have been very happy studying and working here.

My deepest gratitude goes to my advisor Prof. Tomas McKelvey. We started the tradition of our one hour-ish weekly meetings when I did my master thesis project on radar signal processing, and it has been a long journey since then. I really appreciate all the technical discussions we had all these years and enjoyed every trip we took together. You have helped me through so many tough moments with full supports and great patience. Nothing would have been possible without you. I would also like to thank my (almost) co-advisor Prof. Konstantinos Diamantaras. I cherish all the discussions and fun we had, not to mention the wonderful visit in Thessaloniki. I am always impressed by all your awesome research ideas and your ability in finding crazy photo opportunities. I would like to express many many thanks to Prof. S.Y. Kung for being a great advisor and delightful host at Princeton. You always challenge me and give me genius advice. Thanks for including me in the wrapping up of your book writing process. The book is exceptional and I have learned so much from you.

I would like to thank Prof. Jian Yang at Chalmers, who was also my advisor during my master thesis project. You have helped me so much both professionally and personally. Without you, it would not have been the same.

Many thanks to my co-authors and colleagues from Medfield Diagnostics AB and Chalmers University of Technology. Hana Dobsicek Trefna, I really enjoyed working and chatting with you and I will see you soon as we still have some future work to do together (those rotten wood logs ain't gonna detect themselves). I would like to thank Prof. Mikael Persson. Thanks for including me in those interesting and promising projects, such as the Stroke Finder. It has been very nice working with you and you are such a delightful person. Many thanks to my co-authors Andreas Fhager and Stefan Candefjord for the nice collaboration. Also thank you Stefan Kidborg. I have not seen you for a long time, but all the work we have done together is memorable. I still kept the candy basket you gave me (only the basket - the candies are gone). Thank you Ann-Christine Lindbom, Natasha Adler Grønbech, Agneta Kinnander and Madeleine Persson for your efficient replies and assistances all these years.

This work has in part been funded by the Swedish Research Council (Vetenskapsrådet) under the contract number A0462701 which is gratefully acknowledged.

Yinan Yu
Göteborg, 2016

List of Included Publications

PAPER 1

Y. Yu, T. McKelvey and S.Y. Kung, "Kernel SODA: a feature reduction technique using kernel based analysis", *In Proceedings of the 12th International Conference on Machine Learning and Applications (ICMLA)*, pages 72-78, Miami, FL, USA, 2013.

PAPER 2

Y. Yu and T. McKelvey, "Learning hierarchical feature space using CLAss-specific Subspace Multiple Kernel - Metric Learning for classification", *submitted to Journal of Machine Learning Research (JMLR).*

PAPER 3

Y. Yu, T. McKelvey, K.I. Diamantaras and S.Y. Kung, "Enhanced distance subset approximation using class-specific kernel functions for supervised learning", *submitted to Neural Networks and Learning Systems, IEEE Transaction on.*

PAPER 4

Y. Yu and T. McKelvey, "Kernel subspace empirical intersection removal for kernel approximation and classification", *submitted to Neural Networks and Learning Systems, IEEE Transaction on.*

PAPER 5

Y. Yu, T. McKelvey, K.I. Diamantaras and S.Y. Kung, "CLAss-specific Subspace Kernel representations and adaptive margin slack minimization for large scale classification", *accepted for publication in Neural Networks and Learning Systems, IEEE Transaction on.*

Software Package

`https://github.com/yinan16/DeepCLASK`

Other Publications

- **Y. Yu**, K. I. Diamantaras, T. Mckelvey and S.Y. Kung, "Enhanced distance subset approximation using class-specific subspace kernel representation for kernel approximation", *In Proceeding of IEEE International Workshop on Machine Learning for Signal Processing*, Vietri sul Mare, Salerno, Italy, 2016.

- **Y. Yu**, K. I. Diamantaras, T. McKelvey and S.Y. Kung, "Adaptive margin slack minimization in RKHS for classification", *In Proceedings of the 38th IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 2016.

- **Y. Yu**, and T. McKelvey, "A robust subspace classification scheme based on empirical intersection removal and sparse approximation", *Integrated Computer-Aided Engineering*, 22 (1), pages 59-69, 2015.

- **Y. Yu**, T. McKelvey, and S.Y. Kung, "Feature reduction based on Sum-of-SNR (SOSNR) optimization", *In Proceedings of the 39th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6756-6760, 2014.

- **Y. Yu**, K. I. Diamantaras, T. McKelvey and S.Y. Kung, "Multiclass Ridge-adjusted Slack Variable Optimization Using selected basis for fast classification", *In Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, pages 1178-1182, Lisbon, Portugal, 2014.

- **Y. Yu**, "Classification of high dimensional signals with small training sample size with applications towards microwave based detection systems", *Licentiate Thesis*, Chalmers University of Technology, 2013.

- **Y. Yu**, T. McKelvey and S.Y. Kung, "A classification scheme for 'high-dimensional-small-sample-size' data using SODA and ridge-SVM with microwave measurement applications", *In Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

- **Y. Yu** and T. McKelvey, "A unified subspace classification framework developed for diagnostic system using microwave signal", *In Proceedings of the 21st European Signal Processing Conference (EUSIPCO)*, Marrakech, Morocco, 2013.

- **Y. Yu**, K. I. Diamantaras, T. McKelvey and S.Y. Kung, "Ridge-Adjusted Slack Variable Optimization for supervised classification", *In Proceedings of IEEE International Workshop on Machine Learning for Signal Processing*, Southampton, United Kingdom, 2013.

- **Y. Yu** and T. McKelvey, "A subspace learning algorithm for microwave scattering signal classification with application to wood quality assessment", *In Proceeding of IEEE International Workshop on Machine Learning for Signal Processing*, Santander, Spain, 2012.

- **Y. Yu**, J. Yang and T. McKelvey "Compact UWB indoor and through-wall radar with precise ranging and tracking", *International Journal of Antennas and Propagation*, 2012. 2011

- **Y. Yu**, S. Maalik, J. Yang, T. McKelvey, K. Malmström, L. Landen and B. Stoew, "A new UWB radar system using UWB CMOS chip", *In Proceedings of*

the 5th European Conference on Antennas and Propagation (EUCAP), Rome, Italy, 2011.

- M. Persson, A. Fhager, H. D. Trefna, **Y. Yu**, T. McKelvey, G. Pegenius, J-E Karlsson and Mikael Elam, "Microwave-based stroke diagnosis making global prehospital thrombolytic treatment possible", *IEEE Transactions on Biomedical Engineering* 61 (11), pages 2806-2817, 2014.

- Q. Jian, J. Yang, **Y. Yu**, P. Bjorkholmy and T. McKelvey, "Detection of breathing and heartbeat by using a simple UWB radar system", *In Proceedings of the 8th European Conference on Antennas and Propagation (EuCAP)*, pages 3078-3081, The Hague, The Netherlands, 2014.

- S. Candefjord, J. Winges, **Y. Yu**, T. Rylander and T. McKelvey, "Microwave technology for localization of traumatic intracranial bleedings - a numerical simulation study", *In Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1948-1951, Osaka, Japan, 2013.

- R. Rothe, **Y. Yu**, and S.Y. Kung, "Parameter design tradeoff between prediction performance and training time for ridge-SVM", *In Proceeding of IEEE International Workshop on Machine Learning for Signal Processing*, pages 1-6, Southampton, United Kingdom, 2013.

Contents

Contents **ix**

I Introductory Chapters

1 Introduction **1**
1.1 A Machine that Learns . . . 1
1.2 Learning Process . . . 2
1.3 Kernel Techniques . . . 3
1.4 Challenges . . . 6
1.4.1 Bias and Variance . . . 6
1.4.2 Robustness . . . 7
1.4.3 Scalability . . . 8
1.4.4 Hyperparameter Tuning . . . 8
1.5 Conclusion . . . 9

2 Reproducing Kernel Hilbert Space and its Application **11**
2.1 Brief Review on Functional Analysis . . . 12
2.1.1 Hilbert Space . . . 12
2.1.2 Riesz Representation Theorem . . . 16
2.2 Reproducing Kernel Hilbert Space (RKHS) and Feature Maps . . . 17
2.2.1 Reproducing Kernels . . . 17
2.2.2 Kernel Functions . . . 18
2.2.3 Constructing Feature Maps from Kernel Functions . . . 21
2.2.4 Dimension of the Feature Space . . . 22
2.3 Structural Risk Minimization and the Representer Theorem . . . 24
2.3.1 Convergence in RKHS . . . 24
2.3.2 Statistical Learning Theory and Structural Risk Minimization 25
2.3.3 Representer Theorem . . . 28
2.3.4 Kernel Tricks . . . 29
2.4 Conclusion . . . 30

3 Kernel Methods in Practice **33**
3.1 Notations . . . 34
3.2 Kernel Model Selection . . . 34

3.2.1 Cross-Validation . . . 36
3.2.2 Multiple-Kernel (MK) Model . . . 36
3.3 Kernel Approximation . . . 37
3.3.1 Motivation . . . 37
3.3.2 Subspace Model for Kernel Approximation . . . 38
3.3.3 Subspace Estimation . . . 40
3.4 Classification Training . . . 41
3.4.1 Overview . . . 41
3.4.2 Connection to Kernel Approximation: . . . 43
3.4.3 Example: LS-SVM . . . 45
3.5 Conclusion . . . 49

4 Summary of Included Papers **51**
4.1 Kernel Subspace Model . . . 53
4.1.1 Overview . . . 53
4.1.2 Included Papers . . . 53
4.2 CLAss-specific Subspace Kernel (CLASK) Function for Classification 53
4.2.1 Class-specific Subspace Model . . . 53
4.2.2 CLAss-Specific Kernel (CLASK) Function . . . 54
4.2.3 Feature Map . . . 55
4.2.4 Included Papers . . . 55
4.2.5 A Feature Transformation System Using CLASK . . . 57
4.3 Software Package: DeepCLASK . . . 59
4.4 Conclusions . . . 59

5 Future work **61**

References **63**

II Included Papers

Paper 1 Kernel SODA: A Feature Reduction Technique Using Kernel Based Analysis **73**
1 Introduction . . . 73
2 Related Work . . . 74
2.1 Linear Discriminant Analysis (LDA) . . . 74
2.2 Principal Component Analysis (PCA) . . . 74
2.3 Successively Orthogonal Discriminant Analysis . . . 75
3 Theory of kernel SODA . . . 77
3.1 KSODA in Intrinsic Space . . . 77
3.2 KSODA in Empirical Space . . . 78
4 Implementation and approximation . . . 79
4.1 KSODA implementation . . . 79
4.2 Approximation . . . 79

4.3 Data selection and numerical invertibility 80
5 Experimental Results 81
6 Acknowledgment 83

References **87**

Paper 2 Learning Hierarchical Feature Space Using CLAss-specific Subspace Multiple Kernel - Metric Learning for Classification **91**
1 Introduction 91
2 Problem formulation 94
2.1 Distance metric and subspace model for a given kernel function 94
2.2 CLAss-specific Subspace Kernel Functions 99
2.3 The CLAss-Specific Multiple-Kernel model 101
3 Algorithms and implementation 102
3.1 Basis matrix $\mathbf{U}_{c,k}$ 102
3.2 Kernel function k_c 103
3.3 Summary of the Algorithm 104
3.4 Remarks 104
4 Learning Hierarchical CLASMK Feature Network 105
5 Experimental Results 107
5.1 One Layer CLASK-ML and CLASMK-ML Compared to Single Kernel Learning 109
5.2 Multi-Layer CLASMK-ML Compared to Other Multi-Layer MK Techniques 111
5.3 Multi-Layer CLASMK-ML Performance with Respect to the Number of Layers 112
5.4 Visual Examples of the Estimated Weights 113
6 Conclusion 114
7 Appendix 128
7.1 Lemma 2.2 128
7.2 Lemma 2.3 128
7.3 Proof of Theorem 2.2 129

References **131**

Paper 3 Enhanced Distance Subset Approximation using Class-specific Kernel Functions for Supervised Learning **135**
1 Introduction 135
2 Class-specific Kernel Subspace Representation 137
2.1 Notations 137
2.2 Low Rank Approximation 137
2.3 Relation to Subspace Model 138
2.4 Class-Specific Subspace Model 139

2.5 CLAss-Specific Kernel (CLASK) function: Feature Space and Low Rank Approximation . . . 139
3 CLASK Parameter Estimation for Classification . . . 142
4 Algorithm . . . 144
4.1 Description of EDSA . . . 144
4.2 Computational Details and Complexity . . . 150
4.3 Theoretical Analysis for EDSA . . . 150
5 Related work . . . 152
6 Results . . . 153
7 Conclusion . . . 153
8 Appendix . . . 155
8.1 Proof of Lemma 3.1 . . . 155
8.2 Proof of Theorem 5.1 . . . 155
8.3 Proof of Lemma 3.2 . . . 157
8.4 Proof of Lemma 3.3 . . . 157
8.5 Proof of Lemma 3.4 . . . 158

References **161**

Paper 4 Kernel Subspace Empirical Intersection Removal for Kernel Approximation and Classification **167**
1 Introduction . . . 167
2 Related work . . . 169
2.1 Metric Learning . . . 169
2.2 Subspace Classifiers . . . 170
3 Preliminaries . . . 171
3.1 Subspace Data Model . . . 171
3.2 Kernel trick and notations . . . 171
3.3 Empirical Risk and Learnability . . . 172
4 Kernel Empirical Subspace Intersection Removal . . . 173
4.1 Motivation . . . 174
4.2 Analysis . . . 175
4.3 Algorithm . . . 179
4.4 Multiple Class-KESIR (MC-KESIR) . . . 180
4.5 Applications . . . 182
5 Experimental Results . . . 182
5.1 Experiment Setup . . . 182
5.2 Prototype Subspace . . . 183
5.3 Results . . . 183
6 Conclusion . . . 184
7 Acknowledgement . . . 185
8 Appendix . . . 199
8.1 Definition of Subspace Intersection . . . 199
8.2 Proof of Lemma 4.1 . . . 199

8.3 Proof of Lemma 4.2 200
8.4 Proof of Lemma 4.3 200

References **203**

Paper 5 CLAss-specific Subspace Kernel Representations and Adaptive Margin Slack Minimization for Large Scale Classification **209**
1 Introduction 209
2 Part 1: Feature extraction using CLASK 212
2.1 Preliminary 212
2.2 CLAss-specific Subspace Kernel (CLASK) representation . . . 214
2.3 Distance metric and its theoretical bound 215
2.4 CLASK-Approximation 217
3 Part 2: Classification using AMSM 220
3.1 Preliminary 220
3.2 Adaptive Margin Slack Minimization (AMSM) Algorithm . . . 222
3.3 Implementation 225
4 Part 3: A scalable framework using CLASK + AMSM 228
4.1 AMSM risk function evaluation based on CLASK feature extraction 228
4.2 Algorithm CLASK+AMSM for a unit processor 228
4.3 Memory efficient sequential processing (MESP) 229
4.4 Parallelized Sequential Processing (PSP) 230
5 Results and discussion 231
5.1 Evaluation on CLASK 232
5.2 AMSM accuracy: unit processor 234
5.3 Scalable framework 234
5.4 Large Scale Benchmark Datasets 235
6 Conclusion 236
7 Acknowledgement 237
8 Appendix: Multiclassification 237

References **247**

Part I

Introductory Chapters

Chapter 1

Introduction

1.1 A Machine that Learns

Machine learning is a branch of techniques under the subject of Artifical Intelligence (AI) [1, 2]. Given collected data, it explores the possibilities of a machine making "correct" decisions based on a learning process. As opposed to a mere lookup table, a learned machine has the ability to treat the **unseen** data with a certain accuracy, i.e. the ability of "prediction". There are mainly two types of prediction tasks that a machine is trying to accomplish by learning: **classification** and **regression**. Classification refers to identifying labels of categorized objects, such as automatically recognizing the breed of a dog given its picture, and the label information refers to the breed that the dog belongs to. Regression, on the other hand, refers to the task of prediction or forecasting. The only difference is that classification has discrete values as its output, whereas regression produces result that contains continuous values. Classification and regression are intertwined. For instance, a classification rule usually involves learning a function with a continuous output and applying a threshold to obtain the discrete value for the classifier. Hence, classification problems can naturally be represented by regression models. By the same principle, a regression problem can be modified into classification by quantizing the desired output. Techniques for these two tasks are interchangeable with minor modifications.

A flowchart illustrating the steps it takes for a machine to complete its mission can be found in Fig. 1.1. Data acquisition is the first step into our data adventure. It usually refers to the actions that a user takes to retrieve and manipulate data from a database. After the query and formatting, an initialization called preprocessing is applied to clean up data and construct a preliminary feature space. This is a domain specific process that requires knowledge of information extraction for applications from different disciplines, such as speech analysis, image processing, medical applications, text sentiment analysis, etc. The output of the preprocessing are called the preliminary features, which are then fed into the next step for further processing. Feature selection [3, 4] is a step that selects a subset of the preliminary features by removing redundant information, such that the computational complexity and storage requirement can be reduced. Of course, this subset selection intends to preserve

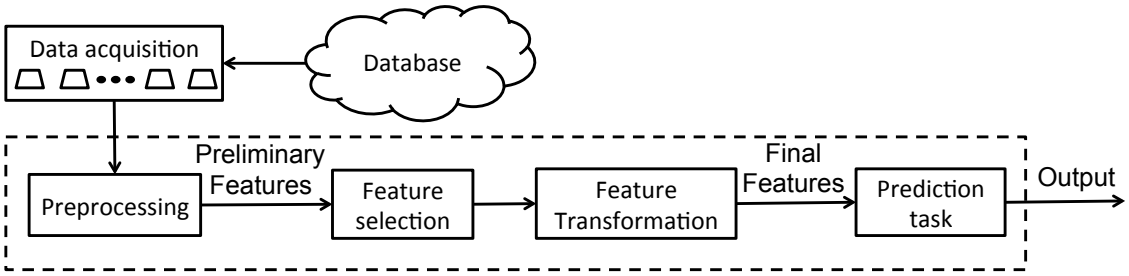

Figure 1.1: A framework for learning a classification rule.

the informative structure of the original data. A subsequential block after feature selection is the feature transformation step, where instead of choosing a subset, the features are transformed onto a high dimensional space and then possibly restricted to a subspace of lower dimension. The purpose of this transformation is to create a new feature space by a linear or nonlinear combination of the selected features. The resulting features are then used as the input for the prediction task as illustrated in the previous paragraph.

Given different machine learning frameworks, Fig. 1.1 is modified accordingly. For instance, the Convolutional Neural Networks (CNN), one of the most popular deep learning techniques for image processing, integrates the three blocks ("Preprocessing", "Feature selection" and "Feature transformation") into one procedure to determine final features using a deep network. On the other hand, the classic Artificial Neural Networks (ANN) has a learning structure such that feature selection, feature transformation and the classifier are all implemented as a part of the backpropagation algorithm. Some other techniques, such as the Principal Component Analysis (PCA) with sparse representations [5, 6], perform feature selection and transformation simultaneously.

1.2 Learning Process

In order to construct a machine that automatically performs prediction on its own as shown in Fig. 1.1, a learning process is needed for each block. Before the learning begins, We need to determine the following components:

- **Model selection:** It refers to choosing an appropriate family of learning models for a given machine learning task. In other words, the learning space is restricted to the selected model family. The learning model contains unknown parameters, which need to be estimated during the learning process. Moreover, model selection typically involves selecting *hyperparameters* within the model family, which are the parameters that one needs to know in advance to be able to solve for the unknowns. For instance, the number of principal components in PCA, the number of hidden layers and units in a neural net, the prior information in a Bayesian framework, etc. These hyperparameters are usually determined manually by the designer, but they can also be automatically selected using

machine learning techniques as well. This is sometimes called "metalearning", since it is "learning" for the learning process.

- **Learning objective:** It is the mathematical formulation of the goal that the machine is expected to achieve. Typically, it is presented as an optimization problem, such as minimizing the squared error in linear regression. Constraints are often applied to further restrict the searching space in addition to the selected family of learning models. As an example, the learning objective of Support Vector Machines (SVM) is to find a hyperplane that maximizes the "margin" of the separation on the training data.

- **Searching algorithm:** Given the learning model and the objective with some constraints, the searching algorithm estimates the unknown parameters in the learning model. The formulation of the learning objective heavily affects the efficiency and optimality of the searching process. For example, if an optimization problem is impossible to solve in polynomial time, or suffers from local optima, searching for the exact solution would be extremely time consuming and difficult. On the other hand, if the learning objective is a convex problem, various existing tools would be available for finding the global optimal solution. Hence, tradeoffs are often being made while designing machine learning algorithms.

The learning process can be categorized into *supervised learning* and *unsupervised learning*. The goal of *supervised learning* is to determine the intrinsic data structure and learning rules given both data and their labels for classification tasks or correct numerical outputs for regression. On the other hand, *unsupervised* learning does not require the label information [2]. Unsupervised learning refers to techniques such as k-means for automatic data clustering [7, 8].

Since the labels are visible to the machine for supervised learning, the parameter estimation is also called *training*, as it resembles the learning process trained by a supervisor. In this case, the data-label pairs are called *training data*. This thesis work focuses on the development of supervised learning techniques.

1.3 Kernel Techniques

The kernel method [9, 10, 11, 12] is one of the most popular nonlinear learning techniques, which is based upon the construction of a high (possibly infinite) dimensional feature space endowed with an **inner product**. Learning techniques are applied in this constructed feature space accordingly. By using kernel techniques, the advantage is twofold:

- The high dimensionality of the constructed feature space provides the possibility of describing complex data structures;

- All computations are carried out by the **kernel function** on the original data. In other words, no explicit computations are required in the high dimensional feature space.

A Demonstration Using Kernel SVM

To gain a better intuition, we demonstrate kernel methods using the Support Vector Machine (SVM), which is also known as the maximal margin classifier. We try to limit this introduction to its minimum. One can find more detailed descriptions from, e.g. [13, 14, 11], and the references therein.

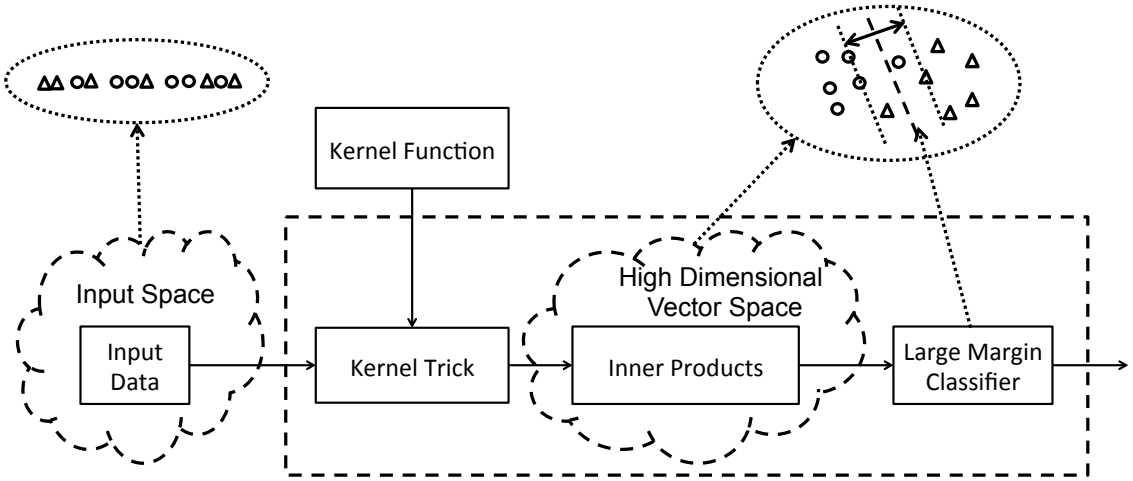

Figure 1.2: A flowchart of kernel SVM.

Soft Margin SVM: First, let us leave kernels aside and focus on the formulation of SVMs. In a classification problem, a SVM tries to find a hyperplane that separates data from two classes with the largest margin. There are two types of margins: the *hard margin* and the *soft margin*. Different margins reflect different concerns, which result in their own distinct formulations. Briefly speaking, a hard margin can only be used when training data are linearly separable, whereas soft margin allows some misclassified data points during training. In fact, soft margin is more commonly used in practice due to its robustness [15].

In Fig. 1.2, a soft margin is illustrated in the picture above the "Large Margin Classifier". It is defined as the distance between the dashed line (the separating hyperplane) and the dotted line (the marginal hyperplane). Mathematically speaking, this distance can be computed using the length of the perpendicular vector (usually denoted as $\mathbf{w}$) that connects these two lines, i.e. $\frac{1}{\|\mathbf{w}\|_2}$. Due to the symmetry of the two marginal hyperplane and also for optimization convenience, one uses $\frac{2}{\|\mathbf{w}\|_2^2}$ to denote the margin, i.e. the distance between the two marginal hyperplanes. In other words, every vector $\mathbf{w}$ defines a hyperplane that separates the data space into two disjoint half-spaces.

Now, given a training set

$$\left\{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathbb{R}^p; y_i \in \{-1, +1\}; i = 1, \cdots, N; p, N \in \mathbb{N}^+\right\},$$

and a nonlinear map: $\varphi : \mathbf{x}_i \mapsto \boldsymbol{\varphi}_i \in \mathbb{R}^q$. the objective is to find a vector $\mathbf{w} \in \mathbb{R}^q$ and

a bias $b \in \mathbb{R}$ such that [13]:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize:}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \eta \sum_j \xi_j \\ \text{subject to:} \quad & \left(\mathbf{w}^T \boldsymbol{\varphi}_i + b\right) y_i \geq 1 - \xi_i, \ \forall i \\ & \xi_i \geq 0, \ \forall i \end{aligned} \tag{1.1}$$

where ξ_i is called the slack variable, which indicates the signed distance from a misclassified data point to the marginal hyperplane. The scalar η is chosen by the user to control the misclassification rate. Generally speaking, the smaller η is, the larger number of misclassified data are allowed during training.

More precisely, a given data pair $(\boldsymbol{\varphi}, y)$ is on the marginal hyperplane if and only if $\mathbf{w}^T\boldsymbol{\varphi} + b = \begin{cases} +1, & \text{for } y = +1 \\ -1, & \text{for } y = -1 \end{cases}$. Therefore, a unified expression for the marginal hyperplane is $\left(\mathbf{w}^T\boldsymbol{\varphi} + b\right) y = 1$. In fact, for any $(\boldsymbol{\varphi}, y)$, the larger $\left(\mathbf{w}^T\boldsymbol{\varphi} + b\right) y$ is, the better this data point is separated from the other class and the signed distance between the marginal hyperplane and $(\boldsymbol{\varphi}, y)$ is expressed by $1 - \left(\mathbf{w}^T\boldsymbol{\varphi} + b\right) y$. Given this analysis, the soft margin SVM can be interpreted as maximizing the margin, while minimizing the distance from the marginal hyperplane to the misclassified training data, i.e. data that satisfy $1 - \left(\mathbf{w}^T\boldsymbol{\varphi}_i + b\right) y_i \geq 0$.

Duality: To solve the optimization problem in Eq. (1.1), one possibility is to analyze its duality. By computing the Lagrangian [16] and its KKT condition, we obtain the dual of Eq. (1.1):

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize:}} \quad & \sum_j \alpha_j - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\varphi}_i^T \boldsymbol{\varphi}_j \\ \text{subject to:} \quad & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \eta, \ \forall i \end{aligned} \tag{1.2}$$

where $\boldsymbol{\alpha}$ is the vector that contains the Lagrangian multipliers and α_i denotes the i^{th} element of $\boldsymbol{\alpha}$. Moreover, the KKT condition results in the following relation between the normal vector $\mathbf{w}$ and Lagrangian multipliers:

$$\mathbf{w} = \sum_i \alpha_i y_i \boldsymbol{\varphi}_i \tag{1.3}$$

In other words, the normal vector $\mathbf{w}$ can be written as a linear combination of training vectors $\boldsymbol{\varphi}_i$'s.

Kernel Trick: When the dimension of $\boldsymbol{\varphi}_i$'s is high, the computation of $\boldsymbol{\varphi}_i^T \boldsymbol{\varphi}_j$ can be prohibitive. In kernel techniques, one reduces the computational complexity by a technique called the "kernel trick".

Let $\mathbf{x}_i, \mathbf{x}_j$ be any data points sampled from the low dimensional input space. Now let us define a "kernel function" k, such that

$$k(\mathbf{x}_i, \mathbf{x}_j) \triangleq \boldsymbol{\varphi}_i^T \boldsymbol{\varphi}_j, \tag{1.4}$$

which means that the computation $\boldsymbol{\varphi}_i^T \boldsymbol{\varphi}_j$ can be carried out in the low dimensional input space $\mathbb{R}^p$. Eq. (1.4) is called the kernel trick in the literature. Briefly speaking, the kernel trick utilizes the kernel function k to produce computations in the high dimensional space, where the classifier is established. This is an efficient replacement since inner products $\boldsymbol{\varphi}_i^T \boldsymbol{\varphi}_j$ are all the computations we need in the high dimensional space for SVM to estimate the unknown parameters α_i's.

Furthermore, when applying the classifier to an unseen data vector $\tilde{\mathbf{x}}$, one has to evaluate the following:

$$f(\tilde{\mathbf{x}}) \triangleq \mathbf{w}^T \varphi(\tilde{\mathbf{x}}) = \sum_i \alpha_i y_i \underbrace{\boldsymbol{\varphi}_i^T \varphi(\tilde{\mathbf{x}})}_{\text{kernel trick}} = \sum_i \alpha_i y_i k(\mathbf{x}_i, \tilde{\mathbf{x}}) \tag{1.5}$$

such that the estimated label $\hat{y}$ for $\tilde{\mathbf{x}}$ is computed as:

$$\hat{y} = \begin{cases} +1, & f(\tilde{\mathbf{x}}) \geq 0 \\ -1, & f(\tilde{\mathbf{x}}) < 0 \end{cases} \tag{1.6}$$

where again, no high dimensional computation is needed in Eq. (1.5).

To summarize, in kernel techniques, the original low dimensional data are mapped onto a high dimensional feature space using the kernel function to gain better flexibility, whereas no explicit high dimensional computation is needed thanks to the kernel trick.

1.4 Challenges

When designing a machine learning system, the selection of the most appropriate family of learning models is not a trivial task. Various trade-offs need to be taken into consideration.

1.4.1 Bias and Variance

Given a learning system, estimations for unknown parameters vary with respect to different training sets. The bias and variance can be computed accordingly for estimated parameters and/or the prediction output. Empirically, the bias and variance can be approximated using the sample mean and the sample variance, respectively. A demonstration can be found in Fig. 1.3. Generally speaking, the trade-off between bias and variance is related to the model complexity. That is, learning models with high complexity tend to have low bias but high variance, and vice versa. Hence, the trade-off can be found by choosing a model family that is suitable for the problem at hand [17].

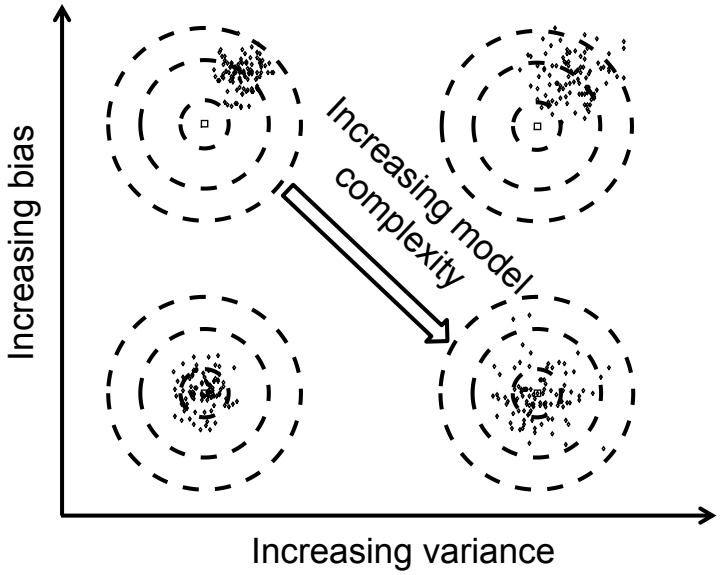

Figure 1.3: A visualization of bias/variance with increasing model complexity. The example shows the estimation of the expected value for some given training sets. With increasing complexity, the bias of the estimate decreases whereas the variance will increase.

Example 1.1 (Kernel function and bias-variance trade-off for SVMs)**.** *As shown in Eq.* (1.4)*, the kernel function* $k(\cdot,\cdot)$ *defines an inner product as a nonlinear function of the input data. This is equivalent to constructing a nonlinear map* $\varphi(\mathbf{x})$*, such that* $\varphi : \mathbf{x} \mapsto \boldsymbol{\varphi}$ *for any input data* $\mathbf{x}$*, where* $\boldsymbol{\varphi}$ *is a high dimensional vector. Hence, by using the kernel function* $k(\cdot,\cdot)$*, the higher dimensional* $\boldsymbol{\varphi}$ *is, the higher complexity the corresponding SVM has. In other words, when working with SVMs associated with "simple" kernel functions* [1]*, one expect high bias with low variance, and vice versa.*

1.4.2 Robustness

There are mainly two aspects when it comes to the robustness of machine learning algorithms:

- Overfitting: It refers to the problem of a learning system lacking generalization ability, which is indicated by achieving a low fitting error on the training data but suffering from inaccurate prediction on unseen testing data.
- Noise/error handling: Given a learning system, the prediction accuracy will vary with increasing noise level or measurement error. The robustness from this perspective can be interpreted as the smoothness and stability of the system.

Example 1.2 (Soft margin SVM and its robustness)**.** *First, we introduce the hard margin SVM. A hard margin SVM is applied when the training data are linearly*

[1] Here a simple kernel function refers to a kernel function corresponding to a low dimension of $\boldsymbol{\varphi}$. Detailed description can be found in Chapter 2

separable. Intuitively speaking, one attempts to find a separating hyperplane f characterized by its normal vector $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i)$, such that i) f categorize all training data to their correct classes; ii) α_i's are only nonzero for a subset of training data. Members in this subset is called "Support Vectors"*. More precisely, the support vectors are the data points located on the marginal hyperplane. However, a hard margin does not take into consideration the presence of noise. It completely depends on a specific subset of training data. In particular, if the training size is small, hard margin SVMs are especially prone to overfitting. A soft margin SVM, on the other hand, finds a separating hyperplane in a weighted consensus fashion. That is, if a well classified data point is "far away" from the marginal hyperplane, its contribution to constructing the classifier is small, whereas the more "ambiguous" a data point is, the higher weight it has. The weighting is controlled by the hyperparameter η in Eq.* (1.1)*. Therefore, in the soft margin SVM, values for α_i's are more smooth compared to the hard margin SVM, which introduce better robustness to the learning process.*

1.4.3 Scalability

The scalability in machine learning usually refers to the computational complexity and/or the storage requirement for training, which can be improved by the following:

1) using approximations instead of exact solutions;

2) choosing the most suitable programming language with efficient coding scheme;

3) taking advantage of parallel/distributed frameworks;

4) exploiting appropriate data structures;

5) finding the trade-off between computational complexity and storage usage.

Example 1.3 (Scalability of SVM)**.** *From the formulation of the soft margin SVM (c.f. Eq.* (1.2)*, there are mainly three concerns regarding the scalability that one should take into account:*

- *Different computational complexity caused by different kernel functions;*

- *Kernel matrix for large training size;*

- *Training time with respect to η for large training size N.*

1.4.4 Hyperparameter Tuning

Hyperparameters refer to the parameters in the learning model that is not part of the variables that the training algorithm is solving for. Typically, their values need to be set manually prior to the learning process. Hyperparameter tuning can be considered as an outer training loop and it needs to be validated by unseen testing data. Generally speaking, the more hyperparameters there is, the more flexible the model

is due to the high degrees of freedom. However, one needs to be cautious that high flexibility often implies high complexity, which might lead to high variance. Moreover, when the number of hyperparameters is large, the evaluation of their quality requires high computational complexity and its "optimality" typically lacks theoretical justification. Hence, one needs to be aware of the trade-off between the flexibility and the number of hyperparameters.

1.5 Conclusion

In this chapter, we have given a brief introduction on the subject of machine learning, where the learning process is decomposed into three components: 1) model selection, 2) learning objective and 3) the searching algorithm. Kernel method is then introduced as one of the most popular nonlinear learning techniques and the main concept is illustrated using the Support Vector Machines. Moreover, we have identified challenges and trade-offs when designing kernel techniques, which gives rise to the potential areas for further development.

Chapter 2

Reproducing Kernel Hilbert Space and its Application

In this chapter, the theoretical background of kernel techniques is introduced. We start with the preliminaries for the definition of the Reproducing Kernel Hilbert Space and its properties, which leads to its applications for solving machine learning problems.

Given an input space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a non-empty input set that consists of any objects, such as numerical vectors, texts, images, DNA strings, pages of websites, etc. The set $\mathcal{Y}$ typically contains scalars that takes either discrete values for classification, or continuous values for regression. The underlying assumption is that the intrinsic distribution of such random objects is "well" represented in a high dimensional feature space $\mathcal{F}$ that can be obtained by a nonlinear feature map $\varphi : \mathcal{X} \to \mathcal{F}$.

The effectiveness of such transformations mainly results from the increased dimensionality, i.e. $dim(\mathcal{F}) \gg dim(\mathcal{X})$. To illustrate, let us consider the classification problem shown in Fig. 2.1, where the "wishful" results is achieved by applying a nonlinear transformation $\varphi(\cdot)$ to a one dimensional input space, such that feature vectors from different classes in the resulting feature space $\mathcal{F} = \mathbb{R}^2$ are "better" separated.

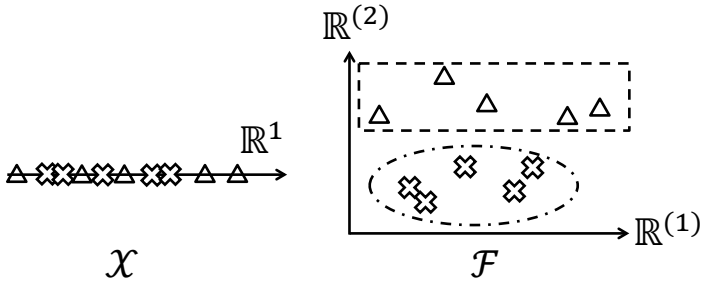

Figure 2.1: A demonstration of the nonlinear mapping $\varphi : \mathcal{X} \to \mathcal{F}$. The crosses and triangles represent data vectors from two different classes. In order to perform classification, the class separability, visualized by the Euclidean distance between the two clusters, is clearly better in the high dimensional space $\mathcal{F}$.

This introductory chapter is organized as follows. After a brief introduction on

some fundamental concepts and definitions in Sec. 2.1, we illustrate the theory behind reproducing kernels and the associated feature spaces. Essentially, the message of Sec. 2.2 is that for every positive definite kernel function there exists a unique Reproducing Kernel Hilbert Space (RKHS), where the inner product is well defined. There also exist other feature spaces that are isometrically isomorphic to the RKHS associated with the same kernel function. Examples of commonly used kernel functions and the construction of feature spaces are then demonstrated. Given the core concepts of kernel techniques, practical issues on optimization in feature spaces are addressed in Sec. 2.3.

2.1 Brief Review on Functional Analysis

In this section, we give a brief introduction on the subject of Reproducing Kernel Hilbert Space (RKHS). An inner product space induces a metric space, which is endowed with a corresponding distance metric used as a similarity measure. In kernel techniques, one constructs an inner product space $\mathcal{H}$ with some extra properties and uses $\mathcal{H}$ as the feature space for learning. Typically, the dimensionality of $\mathcal{H}$ is very large or even infinite. A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is applied to replace the explicit computation of inner products in $\mathcal{H}$. In fact, the space $\mathcal{H}$ is completely characterized by the kernel function, which allows us to work in the original space $\mathcal{X}$ instead of the high dimensional space $\mathcal{H}$. Since operations are not directly carried out in the RKHS, the properties of $\mathcal{H}$ is rather unclear to us, especially in the infinite dimensional case. Hence, to gain better understanding, one has to borrow some tools from mathematical analysis.

This section is fairly self-contained, but we assume a basic familiarity with the fundamental concepts in functional analysis, such as metric space, normed vector space, completeness, etc. Throughout this presentation, the field of the scalars for the vector space is the field of real numbers $\mathbb{R}$.

2.1.1 Hilbert Space

One of the most important concept for a vector space is its **basis**.

Definition 2.1 (Basis)**.** A basis of a (finite or infinite dimensional) vector space $\mathcal{V}$ is a linearly independent subset $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \cdots\}$ of vectors that span $\mathcal{V}$. Equivalently, a subset $\{\mathbf{e}_1, \mathbf{e}_2, \cdots\}$ is a basis if and only if every $\mathbf{v} \in \mathcal{V}$ can be uniquely written as

$$\mathbf{v} = a_1\mathbf{e}_{i_1} + \cdots + a_n\mathbf{e}_{i_n}$$

where $a_1, \cdots, a_n \in \mathbb{R}$ for some finite $n \in \mathbb{Z}^+$ and $i_j \in \mathbb{Z}^+$ for $1 \leq j \leq n$.

The definition implies an important property for a subset being a basis: the finiteness in the linear combination for representing any vectors in that space. The existence of a basis for any finite dimensional vector space is obvious, whereas it is less clear in the infinite dimensional case. In fact, by accepting Zorn's Lemma (or equivalently, the axiom of choice), one can claim that every vector space has a basis.

Lemma 2.1 (Zorn's Lemma)**.** *If $\mathcal{X}$ is a partially ordered set and every linearly ordered subset of $\mathcal{X}$ has an upper bound, then $\mathcal{X}$ has a maximal element.*

To show that every vector space has a basis, let $\mathcal{T}$ be a collection of all linearly independent subsets of a vector space $\mathcal{V}$. We then order the elements in $\mathcal{T}$ by inclusion $\mathcal{E}_1 \subset \mathcal{E}_2 \subset \cdots$. Hence, the union of the chain, denoted by $\mathcal{U}$, also belongs to $\mathcal{T}$ due to the definition of $\mathcal{T}$. Now we have found ourselves an upper bound, $\mathcal{U}$. By accepting Zorn's Lemma, we know that there is a *maximal linearly independent* set $\mathcal{M}$ in $\mathcal{T}$. To show $\mathcal{M}$ is a basis for $\mathcal{V}$, we have to show that this maximal set spans $\mathcal{V}$. This is easily proven by contradiction. Assume that there is a element $\mathbf{v} \in \mathcal{V}$, such that $\mathbf{v}$ is linearly independent of all vectors in $\mathcal{M}$, then we have $\mathcal{M} \subset \mathcal{M} \cup \mathbf{v}$, which contradicts the maximality of $\mathcal{M}$.

In particular, when $dim(\mathcal{V})$ is infinity, the basis is called the *Hamel basis*.

While the existence of a basis in any vector space does sound comforting, Zorn's Lemma is not quite useful, i.e. it does not provide any construction of the basis. In fact, there is no practical way of finding a Hamel basis in general. This gives rise to the importance of the *orthonormal basis*, where infinite sums of linearly independent vectors are allowed to represent a vector. Moreover, for the infinite sum to make sense, one needs the definition of convergence on the vector space, which requires the notion of topological vector spaces. This brings us to the definition of the inner product space. Note that we have neglected the definitions of other important objects, such as metric spaces and normed vector spaces, since they are naturally induced by inner product spaces.

Definition 2.2 (Inner Product)**.** Let $\mathcal{H}$ be a vector space. An inner product on $\mathcal{H}$ is a map $(\mathbf{v}, \mathbf{u}) \to \langle \mathbf{v}, \mathbf{u} \rangle_{\mathcal{H}}$ from $\mathcal{H} \times \mathcal{H} \to \mathbb{R}$ such that:

- $\langle a\mathbf{u} + b\mathbf{v}, \mathbf{z} \rangle_{\mathcal{H}} = a\langle \mathbf{u}, \mathbf{z} \rangle_{\mathcal{H}} + b\langle \mathbf{v}, \mathbf{z} \rangle_{\mathcal{H}}$, for all $\mathbf{u}, \mathbf{v}, \mathbf{z} \in \mathcal{H}$ and $a, b \in \mathbb{R}$.
- $\langle \mathbf{v}, \mathbf{u} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle$, for all $\mathbf{u}, \mathbf{v} \in \mathcal{H}$.
- $\langle \mathbf{u}, \mathbf{u} \rangle \in (0, \infty)$, for all nonzero $\mathbf{u} \in \mathcal{H}$.

It is obvious that every inner product induces an associated norm $\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle_{\mathcal{H}}}$ and thus convergence can be defined accordingly. Without ambiguity, unless necessary, we simply denote the inner product and the induced norm using $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively, without specifying the corresponding vector space.

Definition 2.3 (Pre-Hilbert Space)**.** A vector space equipped with an inner product is called an inner product space, or a pre-Hilbert space. One advantage of an inner product space over a normed space is that an inner product space allows us to define orthogonality.

Definition 2.4 (Orthogonality)**.** Given a pre-Hilbert space $\mathcal{H}_0$ and $\mathbf{u}, \mathbf{v} \in \mathcal{H}_0$, we say $\mathbf{u}$ and $\mathbf{v}$ are orthogonal, denoted as $\mathbf{u} \perp \mathbf{v}$, if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.

Some well-known inequalities and identities are listed below. They are frequently applied for various proofs in a pre-Hilbert spaces.

- Cauchy-Schwartz Inequality:

$$|\langle \mathbf{u}, \mathbf{v} \rangle|^2 \leq \langle \mathbf{u}, \mathbf{u} \rangle \cdot \langle \mathbf{v}, \mathbf{v} \rangle \tag{2.1}$$

- Triangle Inequality:

$$|\langle \mathbf{u} + \mathbf{v} \rangle| \leq \|\mathbf{u}\| + \|\mathbf{v}\| \tag{2.2}$$

- The parallelogram law:

$$\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 = \frac{1}{2}\left(\|\mathbf{u} + \mathbf{v}\|^2 + \|\mathbf{u} - \mathbf{v}\|^2\right) \tag{2.3}$$

- The polarization identity:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \frac{1}{4}\left(\|\mathbf{u} + \mathbf{v}\|^2 - \|\mathbf{u} - \mathbf{v}\|^2\right) \tag{2.4}$$

- Pythagorean Theorem: If $\mathbf{u} \perp \mathbf{v}$, then

$$\|\mathbf{u} + \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 \tag{2.5}$$

An inner product space without any extra properties is called a pre-Hilbert space for a reason: a Hilbert space is the "comfort zone" for analysis and to construct a Hilbert space from a pre-Hilbert space, there is just one step missing.

Definition 2.5 (Hilbert Space)**.** A Hilbert space $\mathcal{H}$ is a pre-Hilbert space that is complete with respect to the norm $\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$, for $\mathbf{u} \in \mathcal{H}$.

To understand completeness, one has to recall the definition of a Cauchy sequence.

Definition 2.6 (Cauchy sequence)**.** Given a metric space $(\mathcal{M}, d)$, a sequence $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \cdots$ is called Cauchy if $d(\mathbf{u}_m, \mathbf{u}_n) \to 0$ as $m, n \to \infty$.

Let d be the distance metric induced by the norm $\|\cdot\|_{\mathcal{H}}$, then

Definition 2.7 (Completeness)**.** Completeness means that every Cauchy sequence in $\mathcal{H}$ converges to a limit that is also in $\mathcal{H}$.

Intuitively, a space being incomplete means that it has "holes" in it, i.e. not every seemingly convergent sequence is converging to a point inside the space. Obviously, that is problematic when we try to find a nice space to analyze infinite sums. Nevertheless, the completion does not take much extra effort. That is, to complete a pre-Hilbert space, one simply adds the limit points of sequences that are convergent in that norm. In other words, to reach any point in a Hilbert space, one simply constructs a Cauchy sequence converging to that point.

Note that for the remainder of the chapter, we use $\mathcal{H}$ to denote Hilbert space.

Remark. Any closed subspace of a Hilbert space is itself a Hilbert space.

One important feature of a Hilbert space is that it enables the concept of projections.

Theorem 2.1. *If $\mathcal{M}$ is a closed subspace of $\mathcal{H}$, then $\mathcal{H} = \mathcal{M} \bigoplus \mathcal{M}^{\perp}$; that is, each $\mathbf{u} \in \mathcal{H}$ can be expressed uniquely as $\mathbf{u} = \mathbf{y} + \mathbf{z}$ where $\mathbf{y} \in \mathcal{M}$ and $\mathbf{z} \in \mathcal{M}^{\perp}$. Moreover, $\mathbf{y}$ and $\mathbf{z}$ are the unique elements of $\mathcal{M}$ and $\mathcal{M}^{\perp}$ whose distance to $\mathbf{u}$ is minimal.*

To characterize a Hilbert space and its subspaces, we introduce the definition of a basis for infinite dimensional space.

Definition 2.8 (Orthonormal Set)**.** A subset $\{\mathbf{u}_\alpha\}_{\alpha \in \mathcal{A}}$ of $\mathcal{H}$ is called orthonormal if $\|\mathbf{u}_\alpha\| = 1$ for all α and $\mathbf{u}_\alpha \perp \mathbf{u}_\beta$ whenever $\alpha \neq \beta$, where $\mathcal{A}$ is some index set.

Theorem 2.2. *If $\{\mathbf{u}_\alpha\}_{\alpha \in \mathcal{A}}$ is an orthonormal set in $\mathcal{H}$, the following are equivalent:*

- *(Completeness) If $\langle \mathbf{z}, \mathbf{u}_\alpha \rangle = 0$ for all α, then $\mathbf{z} = 0$.*
- *(Parseval's Identity) $\|\mathbf{z}\|^2 = \sum_{\alpha \in \mathcal{A}} |\langle \mathbf{z}, \mathbf{u}_\alpha \rangle|^2$ for all $\mathbf{z} \in \mathcal{H}$.*
- *For each $\mathbf{z} \in \mathcal{H}$, $\mathbf{z} = \sum_{\alpha \in \mathcal{A}} \langle \mathbf{z}, \mathbf{u}_\alpha \rangle \mathbf{u}_\alpha$, where the sum on the right has only countably many nonzero terms and converges in the norm topology no matter how these terms are ordered.*

Definition 2.9 (Orthonormal Basis)**.** An orthonormal set having the properties in Theorem 2.2 is called an orthonormal basis.

Theorem 2.3 (Separable Hilbert space)**.** *Every Hilbert space has an orthonormal basis and a Hilbert space $\mathcal{H}$ is separable iff it has a countable orthonormal basis, in which case every orthonormal basis is countable.*

When $\mathcal{H}$ is separable (i.e. $\mathcal{H}$ contains a countable dense subset), a standard procedures can be applied to obtain an orthonormal basis. Given $\{\mathbf{x}_n\}_1^\infty$ a linearly independent subset in $\mathcal{H}$, we recall two such methods:

- Gram-Schmidt process
  - Let $\mathbf{u}_1 = \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|}$;
  - Having defined $\{\mathbf{u}_1, \cdots, \mathbf{u}_{N-1}\}$, set $\mathbf{v}_N = \mathbf{x}_N - \sum_{n=1}^{N-1} \langle \mathbf{x}_N, \mathbf{u}_n \mathbf{u}_n \rangle$, and $\mathbf{u}_N = \frac{\mathbf{v}_N}{\|\mathbf{v}_N\|}$;
  - The resulting set $\{\mathbf{u}_1, \mathbf{u}_2, \cdots\}$ is then an orthonormal basis for $\mathcal{H}$.
- Principal Component Analysis

  Let $\mathbf{X}$ be a matrix with $\mathbf{x}_i$ as its i^{th} column, for $0 < i \leq \infty$. A set of orthonormal vectors $\mathbf{u}_1, \mathbf{u}_2, \cdots$ are called the Principal Components of a space $\mathcal{H}$ if $\mathbf{u}_k$ is the k^{th} eigenvector of matrix $\mathbf{X}\mathbf{X}^T$, where indices k's are sorted with respect to eigenvalues of $\mathbf{X}\mathbf{X}^T$ in a descending order.

Note that in the literature, separability is usually assumed for Hilbert spaces. In fact, as Folland [18] has pointed out, "most Hilbert spaces that arise in practice are separable". However, the separability is not a property that a Hilbert space possesses automatically. Interested readers can refer to [18, 19] for more fundamental proofs and discussions.

2.1.2 Riesz Representation Theorem

The Riesz Representation theorem is one of the most fundamental results in functional analysis of Hilbert spaces and it provides tools for proving and analyzing the existence and uniqueness of a RKHS and its associated kernel function.

Definition 2.10 (Linear functional)**.** Given Hilbert space $\mathcal{H}$, a function $T : \mathcal{H} \to \mathbb{R}$ is called a linear functional if

$$T(a\mathbf{u} + b\mathbf{v}) = aT(\mathbf{u}) + bT(\mathbf{v}) \tag{2.6}$$

where $\mathbf{u}, \mathbf{v} \in \mathcal{H}$ and $a, b \in \mathbb{R}$.

Definition 2.11 (Bounded)**.** A linear functional $T : \mathcal{H} \to \mathbb{R}$ is called bounded if there exists $C \geq 0$ such that $|T\mathbf{u}| \leq C\|\mathbf{u}\|_{\mathcal{H}}$, for all $\mathbf{u} \in \mathcal{H}$.

Definition 2.12 (Continuous)**.** If $\mathbf{u} \in \mathcal{H}$, linear functional T is called continuous at $\mathbf{u}$ if for every neighborhood O of $T(\mathbf{u})$ there is a neighborhood U of $\mathbf{u}$ such that $T(U) \subset O$. In particular, the functional T is called continuous iff T is continuous at every $\mathbf{u} \in \mathcal{H}$.

Proposition 2.1. *For a linear functional $T : \mathcal{H} \to \mathbb{R}$, the following are equivalent:*

- *T is continuous on $\mathcal{H}$.*
- *T is continuous at 0.*
- *T is bounded.*

So far we have been talking about linear functional T, the domain $\mathcal{H}$ of T, and T being continuous/bounded.

Definition 2.13 (Dual Space)**.** The space of bounded linear functionals on $\mathcal{H}$ is called the dual space of $\mathcal{H}$ and is denoted by $\mathcal{H}^*$.

Remark. The dual space of a Hilbert space is a Hilbert space itself.

Theorem 2.4 (Riesz Representer Theorem)**.** *If $f \in \mathcal{H}^*$, there is a* **unique** $\mathbf{u} \in \mathcal{H}$*, such that $f(\mathbf{v}) = \langle \mathbf{v}, \mathbf{u} \rangle$ for all $\mathbf{v} \in \mathcal{H}$.*

The Riesz Representation theorem has established the relation between a Hilbert space and its dual through the inner product. It is an essential tool to show the **existence** and **uniqueness** of the kernel function on an RKHS.

The key concepts introduced in this section can be found in Fig. 2.2. To summarize, a Hilbert space is a complete vector space that endowed with an inner product. Completeness secures the analysis. For instance, completeness makes sense of limits in that space. Compared to Banach spaces, which are complete normed vector spaces, Hilbert spaces come with a natural definition of orthogonality and projection. Furthermore, by imposing separability, we can construct a countable orthonormal basis by using the Gram-Schmidt process. The dual space of a Hilbert space is defined as the space of all the continuous linear functionals, which is in fact a Hilbert space itself.

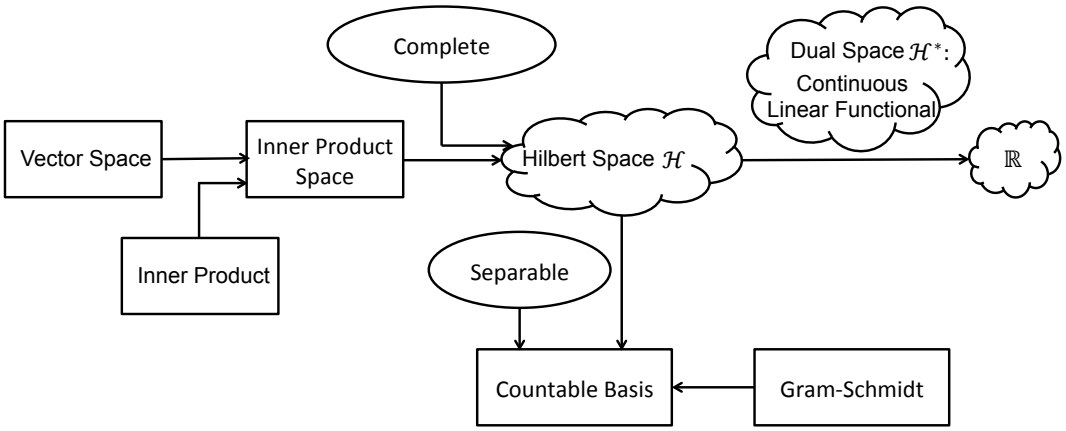

Figure 2.2: A summary of Sec. 2.1.1 and Sec. 2.1.2.

2.2 Reproducing Kernel Hilbert Space (RKHS) and Feature Maps

2.2.1 Reproducing Kernels

Definition 2.14 (Reproducing Kernel Hilbert Space (RKHS))**.** Given a non-empty set $\mathcal{X}$, a Reproducing Kernel Hilbert Space (RKHS) is a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$ with a reproducing kernel whose span is dense in $\mathcal{H}$. Specifically, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a **reproducing kernel** of $\mathcal{H}$ if k satisfies:

- $\forall \mathbf{x} \in \mathcal{X}$, $k(\mathbf{x}, \cdot) \in \mathcal{H}$;
- (The reproducing property) $\forall \mathbf{x} \in \mathcal{X}$, $\langle f, k(\mathbf{x}, \cdot)\rangle = f(\mathbf{x})$, for all $f \in \mathcal{H}$. In particular, $\langle k(\mathbf{x}, \cdot), k(\tilde{\mathbf{x}}, \cdot)\rangle = k(\mathbf{x}, \tilde{\mathbf{x}})$.

The span of the reproducing kernel being dense means that $k(\mathbf{x}, \cdot)$ spans $\mathcal{H}$. Note that we reserve the notation $\mathcal{H}$ to denote a Reproducing Kernel Hilbert Space.

From the Riesz Representation theorem, we know that given $f \in \mathcal{H}$, if there exists a function $k(\mathbf{x}, \cdot) \in \mathcal{H}$, such that $f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot)\rangle$, then $k(\mathbf{x}, \cdot)$ must be unique.

Remark. For a given reproducing kernel k and $\mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X}$, let $\mathcal{H}$ be the RKHS associated with the function k. We have the following clarification on the notions:

- $k(\mathbf{x}, \cdot) : \mathcal{X} \to \mathcal{H}$ is a function of $\mathbf{x}$, i.e. we can define a nonlinear map $\varphi : \mathcal{X} \to \mathcal{H}$, such that $\varphi(\mathbf{x}) = k(\mathbf{x}, \cdot)$. Also, we can view $k(\mathbf{x}, \cdot)$ as a vector in the RKHS.
- $k(\mathbf{x}, \tilde{\mathbf{x}})$ is a scalar, i.e. $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle k(\mathbf{x}, \cdot), k(\tilde{\mathbf{x}}, \cdot)\rangle_{\mathcal{H}} = \langle \varphi(\mathbf{x}), \varphi(\tilde{\mathbf{x}})\rangle_{\mathcal{H}}$, due to the reproducing property.
- k is symmetric in its arguments, i.e. $k(\mathbf{x}, \tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}}, \mathbf{x})$

- Every function $f \in \mathcal{H}$ can be written as a linear combination of feature maps, i.e.

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \cdot) \tag{2.7}$$

where α_i's are coefficients. It directly follows that

$$\begin{aligned} \langle f, k(\mathbf{x}, \cdot)\rangle = & \langle \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \cdot), k(\mathbf{x}, \cdot)\rangle \\ = & \sum_{i=1}^{m} \langle \alpha_i k(\mathbf{x}_i, \cdot), k(\mathbf{x}, \cdot)\rangle \\ = & \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \end{aligned} \tag{2.8}$$

where $m \in \mathbb{N}$ and α_i's are coefficients.

Example 2.1. *Denote* $\mathbf{x}_i = \begin{bmatrix} x_i^1 \\ x_i^2 \end{bmatrix}$ *for any index* i*, let* $k(\mathbf{x}_i, \cdot) = \begin{bmatrix} x_i^1 \\ x_i^2 \\ x_i^1 x_i^2 \end{bmatrix}$*. For all* $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots \in \mathbb{R}^2$ *The span of* $k(\mathbf{x}, \cdot)$ *would be the space spanned by*

$$\{k(\mathbf{x}_1, \cdot), k(\mathbf{x}_2, \cdot), \cdots\cdots\} = \left\{ \begin{bmatrix} x_1^1 \\ x_1^2 \\ x_1^1 x_1^2 \end{bmatrix}, \begin{bmatrix} x_2^1 \\ x_2^2 \\ x_2^1 x_2^2 \end{bmatrix}, \cdots \right\}$$

The representer $k(\mathbf{x}, \cdot)$ *can be interpreted as the nonlinear map* $\varphi : \mathbb{R}^2 \to \mathbb{R}^3$*.*

Definition 2.15 (Positive Definite (PD) kernel)**.** A real-valued symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite (PD) kernel if for all $n \geq 1$, $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathcal{X}$, $c_1, \cdots, c_n \in \mathbb{R}$

$$\sum_{i,j=1}^{n} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \tag{2.9}$$

Theorem 2.5 (Moore-Aronszajn Theorem [20])**.** *To every positive definite function* k *on* $\mathcal{X} \times \mathcal{X}$ *there corresponds a unique RKHS of real valued functions on* $\mathcal{X}$ *and vice versa.*

Theorem 2.5 is one of the most fundamental results in kernel methods. It guarantees the existence and uniqueness for the RKHS and its corresponding kernel function. In other words, by defining a PD kernel function, a unique RKHS is generated automatically. The function norm in the RHKS $\|\cdot\|_{\mathcal{H}}$ is thus induced by the kernel function.

2.2.2 Kernel Functions

As stated above, given a positive definite kernel function, we do not have to worry about the existence of the corresponding RHKS. The question is then which functions are positive definite functions.

Examples of Kernel Functions:

- Linear kernel:
$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{x}^T \tilde{\mathbf{x}} \tag{2.10}$$

- Polynomial kernel:
$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \left(\langle \mathbf{x}, \tilde{\mathbf{x}} \rangle + c\right)^d, \quad c \in \mathbb{R}_0^+, \ d \in \mathbb{Z}^+ \tag{2.11}$$

- RBF kernel:
$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\left(-\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2}{2\sigma^2}\right), \quad \sigma \in \mathbb{R}^+ \tag{2.12}$$

- Sigmoid kernel:
$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \tanh\left(a\mathbf{x}^T \tilde{\mathbf{x}} + b\right), \quad a, b \in \mathbb{R}^+, \ \mathbf{x} \text{ real valued vector} \tag{2.13}$$

- Sinc kernel:
$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{\sin\left(\|\mathbf{x} - \tilde{\mathbf{x}}\|_2\right)}{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2} \tag{2.14}$$

- String kernels [21]: Given an alphabet Σ, a string s is a finite sequence of characters from Σ with cardinality $|s|$. Let u be a subsequence of s, i.e. there exists indices $\{i_{(1)}, \cdots, i_{(|u|)}\}$ sorted in ascending order, such that $u_j = s_{i_{(j)}}$, for $j = 1, \cdots, |u|$, where $|u| < |s|$. Denote $s[i] := u$ and $l(i) := |s[i]|$. Let Σ^n be the set of all finite strings of length n and by Σ^* be the set of all strings
$$\Sigma^* = \sum_{n=0}^{\infty} \Sigma^n$$
The feature mapping for a string s is given by defining the u coordinate $\varphi_u(s)$ for each $u \in \Sigma^n$. Define $\varphi_u(s) = \sum_{i:u=s[i]} \lambda^{l(i)}$, for some $\lambda \leq 1$. Given strings s and t, the kernel function is then defined as:
$$\begin{aligned} k_n(s,t) &= \sum_{u \in \Sigma^n} \langle \varphi_u(s), \varphi_u(t) \rangle \\ &= \sum_{u \in \Sigma^n} \sum_{i:u=s[i]} \lambda^{l(i)} \sum_{j:u=t[j]} \lambda^{l(j)} \\ &= \sum_{u \in \Sigma^n} \sum_{i:u=s[i]} \sum_{j:u=t[j]} \lambda^{l(i)+l(j)} \end{aligned}$$
It turns out that the string kernel can be efficiently evaluated using dynamic programming techniques [21]. One example is shown in Tab. 2.1. The kernel between the work "car" and "cat" is $k(\text{car}, \text{cat}) = \lambda^4$. Since $k(\text{car}, \text{car}) = k(\text{cat}, \text{cat}) = 2\lambda^4 + \lambda^6$, the normalized kernel is:
$$\frac{k(\text{car}, \text{cat})}{\sqrt{k(\text{car}, \text{car})}\sqrt{k(\text{cat}, \text{cat})}} = \frac{1}{2 + \lambda^2}$$

	c-a	c-t	a-t	b-a	b-t	c-r	a-r
φ(cat)	λ^2	λ^3	λ^2	0	0	0	0
φ(car)	λ^2	0	0	0	0	λ^3	λ^2
φ(bat)	0	0	λ^2	λ^3	λ^2	0	0

Table 2.1: An example of string kernel

Combination of Kernel Functions:

Let k_1 and k_2 be two PD kernel functions, then the following functions are also valid PD kernel functions:

- $k(\mathbf{x}, \tilde{\mathbf{x}}) = ck_1(\mathbf{x}, \tilde{\mathbf{x}})$, $c \in \mathbb{R}^+$
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = f(\mathbf{x})k_1(\mathbf{x}, \tilde{\mathbf{x}})f(\tilde{\mathbf{x}})$, where $f : \mathcal{X} \to \mathbb{R}$
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = p\left(k_1(\mathbf{x}, \tilde{\mathbf{x}})\right)$, where $p(\cdot)$ is polynomial with non-negative coefficients
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\left(\frac{k_1(\mathbf{x}, \tilde{\mathbf{x}})}{\sigma^2}\right)$, $\sigma \in \mathbb{R}$
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\left(-\frac{k_1(\mathbf{x}, \mathbf{x}) - 2k_1(\mathbf{x}, \tilde{\mathbf{x}}) + k_1(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})}{\sigma^2}\right)$, $\sigma \in \mathbb{R}$
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = k_1(\mathbf{x}, \tilde{\mathbf{x}}) + k_2(\mathbf{x}, \tilde{\mathbf{x}})$
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = k_1(\mathbf{x}, \tilde{\mathbf{x}})k_2(\mathbf{x}, \tilde{\mathbf{x}})$
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{x}^T\mathbf{A}\tilde{\mathbf{x}}$, where $\mathbf{A}$ is a symmetric PSD matrix
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = \left(k_1(\mathbf{x}, \tilde{\mathbf{x}}) + c\right)^d$, where $c \in \mathbb{R}^+$ and $d \in \mathbb{Z}^+$
- $k(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{k_1(\mathbf{x}, \tilde{\mathbf{x}})}{\sqrt{k_1(\mathbf{x}, \mathbf{x})k_1(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})}}$, normalization of kernel functions

Stationary Kernels $k(\mathbf{x}, \tilde{\mathbf{x}}) = k'(|\mathbf{x} - \tilde{\mathbf{x}}|)$:

A kernel function $k(\mathbf{x}, \tilde{\mathbf{x}})$ is called stationary or translation invariant if it is a function of only the distance between $\mathbf{x}$ and $\tilde{\mathbf{x}}$, i.e. $k(\mathbf{x}, \tilde{\mathbf{x}}) = k'(|\mathbf{x} - \tilde{\mathbf{x}}|)$. For instance, the RBF kernel is stationary, whereas the polynomial kernel is non-stationary. Since a non-stationary kernel function depends on not only the relative relation, but also the position of $\mathbf{x}$ and $\tilde{\mathbf{x}}$, coordinates need to be chosen to find the basis that describe the feature space. The origin is usually chosen to be the mass center of the training set [22].

$$\begin{aligned} k(\mathbf{x}, \tilde{\mathbf{x}}) &\leftarrow \left(\varphi(\mathbf{x}) - \mathbb{E}\left(\varphi(\mathbf{x})\right)\right)^T \left(\varphi(\tilde{\mathbf{x}}) - \mathbb{E}\left(\varphi(\tilde{\mathbf{x}})\right)\right) \\ &= \varphi(\mathbf{x})^T\varphi(\tilde{\mathbf{x}}) - \mathbb{E}\left(\varphi(\mathbf{x})\right)^T \varphi(\tilde{\mathbf{x}}) - \varphi(\mathbf{x})^T\mathbb{E}\left(\varphi(\tilde{\mathbf{x}})\right) + \mathbb{E}\left(\varphi(\mathbf{x})\right)^T \mathbb{E}\left(\varphi(\tilde{\mathbf{x}})\right) \end{aligned}$$

where $\mathbb{E}(\varphi(\cdot))$ can be estimated using the training data $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$.

$$k(\mathbf{x}, \tilde{\mathbf{x}}) \leftarrow k(\mathbf{x}, \tilde{\mathbf{x}}) - \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}) - \frac{1}{N}\sum_{i=1}^{N} k(\mathbf{x}_i, \tilde{\mathbf{x}}) + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.15}$$

More information on the classification and properties of kernel functions can be found in [23, 24].

2.2.3 Constructing Feature Maps from Kernel Functions

For a given PD kernel function k, the existence and uniqueness of the RKHS is guaranteed. However, the construction of RKHS is not always convenient in practice. Fortunately, there exist other features maps $\varphi : \mathcal{X} \to \mathcal{F}$ for $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \varphi(\mathbf{x}), \varphi(\tilde{\mathbf{x}}) \rangle_{\mathcal{F}}$, such that the inner product is preserved in the sense that $\mathcal{H}$ and $\mathcal{F}$ are associated with the same kernel function. Note that not every feature space is RKHS.

Example 2.2 (Feature spaces are not unique)**.** *Given any* $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^2$ *with* $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ *and* $\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}$*, let* $k(\mathbf{x}, \tilde{\mathbf{x}}) = x_1\tilde{x}_1 + x_2\tilde{x}_2 + 2x_1x_2\tilde{x}_1\tilde{x}_2$*, we can construct the following feature maps:*

- $\varphi_1(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & \sqrt{2}x_1x_2 \end{bmatrix}^T$
- $\varphi_2(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_1x_2 & x_1x_2 \end{bmatrix}^T$

such that $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \varphi_1(\mathbf{x}), \varphi_1(\tilde{\mathbf{x}}) \rangle = \langle \varphi_2(\mathbf{x}), \varphi_2(\tilde{\mathbf{x}}) \rangle$.

Nevertheless, feature space $\mathcal{F}$ and RKHS $\mathcal{H}$ are closely related, since they share the same kernel function k. More precisely,

Definition 2.16 (Isomorphism)**.** Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ are said to be isometrically isomorphic if there is a **linear bijective** map $T : \mathcal{H}_1 \to \mathcal{H}_2$ such that $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{H}_1} = \langle T\mathbf{u}, T\mathbf{v} \rangle_{\mathcal{H}_2}$

Isomorphism preserves inner product between two Hilbert spaces. Clearly, all feature spaces associated with the same kernel function are isometrically isomorphic. In this section, we introduce three methods to construct the feature maps $\varphi : \mathcal{X} \to \mathcal{F}$: 1) the RKHS map; 2) the Mercer map and 3) the kernel matrix decomposition.

- RKHS map:

  Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive kernel function. According to Theorem 2.5, every positive definite kernel is associated with a unique RKHS $\mathcal{H}$. The feature map is defined as $k(\mathbf{x}, \cdot)$.

- Mercer Map:

  Another way of constructing the feature map is using the Mercer Theorem:

Theorem 2.6 (Mercer Theorem)**.** *Let $\mathcal{X}$ be compact and k a positive semidefinite kernel function that is continuous and satisfies*

$$\int_{\mathbf{x}}\int_{\mathbf{y}} k^2(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} < +\infty$$

Then there exists $\lambda_1 \geq \cdots \lambda_2 \geq \cdots \geq 0$ and functions $\left\{\psi_i(\cdot)\right\}$, such that

$$k(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x})\psi_i(\mathbf{y}), \quad \forall \mathbf{x},\mathbf{y} \in \mathcal{X}$$

The Mercer map is then defined as

$$\varphi(\mathbf{x}) = \begin{bmatrix} \sqrt{\lambda_1}\psi_1(\mathbf{x}) \\ \sqrt{\lambda_2}\psi_2(\mathbf{x}) \\ \vdots \end{bmatrix} \tag{2.16}$$

- Kernel matrix decomposition:

  Given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a finite set $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, let $\mathbf{K} \in \mathbb{R}^{N\times N}$, where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $1 \leq \forall i,j \leq N$. Matrix $\mathbf{K}$ is a PSD matrix by definition. Let $\mathbf{K} = \mathbf{U}\mathbf{S}\mathbf{U}^T$ using the Singular Value Decomposition, where $\mathbf{U} = \left[\mathbf{u}_1, \cdots, \mathbf{u}_N\right]^T$ and $\mathbf{S} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ 0 & \cdots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_r \end{bmatrix}$, $\lambda_i > 0$. The feature vectors can be constructed as

$$\varphi(\mathbf{x}_i) = k(\mathbf{x}_i, \cdot) = \mathbf{S}^{1/2}\mathbf{u}_i \tag{2.17}$$

2.2.4 Dimension of the Feature Space

In this section, we show some examples to illustrate some nonlinear mappings $\varphi(\cdot)$ and their dimensionality.

Given $\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^2$, where $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}$. Let us expand the expressions for various kernel functions and compare them with the explicit inner product $\varphi(\mathbf{x})^T\varphi(\tilde{\mathbf{x}})$.

- Polynomial kernel $k(\mathbf{x},\tilde{\mathbf{x}}) = \left(1 + \mathbf{x}^T\tilde{\mathbf{x}}\right)^d$ (with degree $d = 2$):
  - Inner product via the kernel function:

$$\begin{aligned} k(\mathbf{x},\tilde{\mathbf{x}}) &= \left(1 + \mathbf{x}^T\tilde{\mathbf{x}}\right)^2 \\ &= \left(1 + \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}\right)^2 \\ &= 1 + 2\left(x_1\tilde{x}_1 + x_2\tilde{x}_2\right) + \left(x_1\tilde{x}_1 + x_2\tilde{x}_2\right)^2 \\ &= 1 + 2\left(x_1\tilde{x}_1 + x_2\tilde{x}_2\right) + x_1^2\tilde{x}_1^2 + x_2^2\tilde{x}_2^2 + 2x_1\tilde{x}_1x_2\tilde{x}_2 \end{aligned}$$

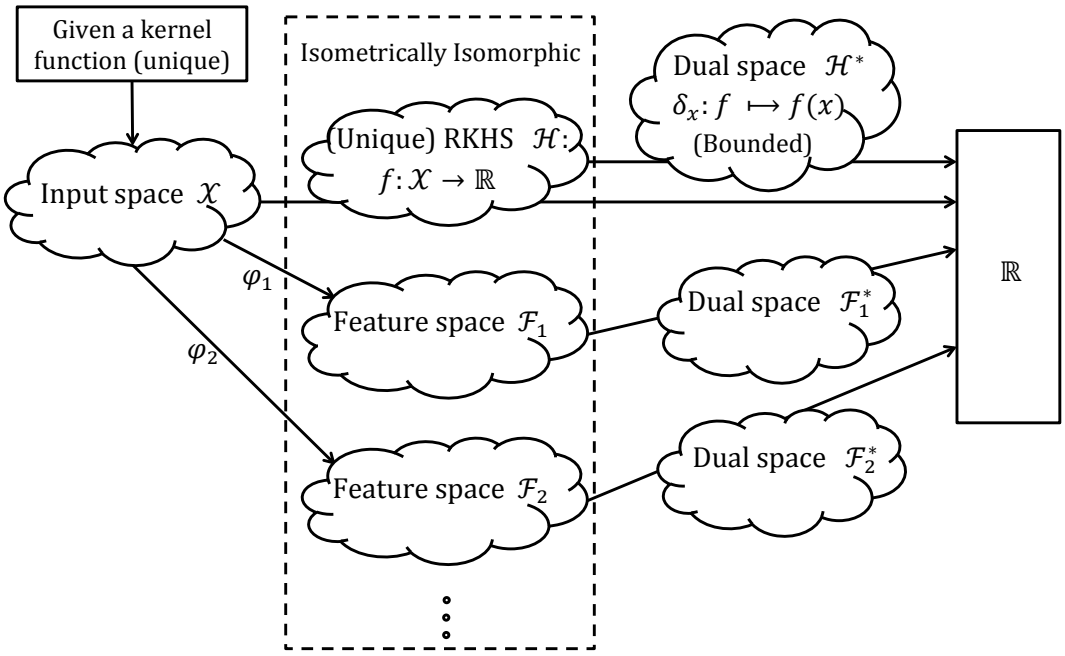

Figure 2.3:

– Explicit inner product:
For any $\mathbf{z} \in \mathbb{R}^2$, the nonlinear mapping associated with polynomial kernel ($d = 2$) is defined as:

$$\varphi(\mathbf{z}) = \begin{bmatrix} z_1^2 & z_2^2 & z_1 z_2 & z_2 z_1 & \sqrt{2} z_1 & \sqrt{2} z_2 & 1 \end{bmatrix}^T$$

Then we have the expression:

$$\varphi(\mathbf{x})^T \varphi(\mathbf{x}) = x_1^2 \tilde{x}_1^2 + x_2^2 \tilde{x}_2^2 + 2 x_1 \tilde{x}_1 x_2 \tilde{x}_2 + 2 x_1 \tilde{x}_1 + 2 x_2 \tilde{x}_2 + 1$$

We have shown an example of the explicit mapping $\varphi(\cdot)$ and have verified that $k(\mathbf{x}, \tilde{\mathbf{x}}) = \varphi(\mathbf{x})^T \varphi(\tilde{\mathbf{x}})$. Given the polynomial kernel with degree 2, the dimension of the new feature space $dim(\mathcal{H}) = 7$.

- RBF kernel $k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\left(-\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|}{2\sigma^2}\right)$:

We can expand the RBF kernel into a Taylor series [25]:

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\left(-\frac{\|\mathbf{x}\|}{2\sigma^2}\right) \left(\sum_{k=1}^{\infty} \frac{1}{k!} \left(\frac{\mathbf{x}^t \tilde{\mathbf{x}}}{\sigma^2}\right)^k\right) \exp\left(-\frac{\|\tilde{\mathbf{x}}\|}{2\sigma^2}\right) \tag{2.18}$$

Let

$$\begin{aligned} k(\mathbf{x}, \tilde{\mathbf{x}}) &= \varphi(\mathbf{x})^T \varphi(\tilde{\mathbf{x}}) \\ &= \begin{bmatrix} \boldsymbol{\varphi}_1 & \cdots & \boldsymbol{\varphi}_{dim(\mathcal{H})} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\varphi}}_1 \\ \vdots \\ \tilde{\boldsymbol{\varphi}}_{dim(\mathcal{H})} \end{bmatrix} = \sum_{k=1}^{dim(\mathcal{H})} \boldsymbol{\varphi}_k \tilde{\boldsymbol{\varphi}}_k. \end{aligned}$$

Eq. (2.18) indicates that $dim(\mathcal{H}) = \infty$.

2.3 Structural Risk Minimization and the Representer Theorem

Given the input space $\mathcal{X} \times \mathcal{Y}$, a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and its associated RKHS $\mathcal{H}$, the task of kernel techniques is to find an optimal function $f \in \mathcal{H}$, such that $f(\mathbf{x})$ can always approximate y accurately, for all pairs $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$.

2.3.1 Convergence in RKHS

Definition 2.17. (Evaluation functional) Given a non-empty set $\mathcal{X}$ and $\mathbf{x} \in \mathcal{X}$. Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. A map $\delta_x : \mathcal{H} \to \mathbb{R}$ is called an evaluation functional if $\delta_x : f \mapsto f(\mathbf{x})$.

It readily follows that evaluation functional is linear, since $\delta_x(a_1h_1 + a_2h_2) = (a_1h_1 + a_2h_2)(\mathbf{x}) = a_1h_1(\mathbf{x}) + a_2h_2(\mathbf{x}) = a_1\delta_x(h_1) + a_2\delta_x(h_2)$, for $h_1, h_2 \in \mathcal{H}$ and $a_1, a_2 \in \mathbb{R}$.

Given this concept, an alternative definition of RKHS is given as follows to show another interesting property of RKHS, i.e. convergence in f implies convergence of $f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$.

Definition 2.18 (Reproducing Kernel Hilbert Space (via evaluation functional))**.** A RKHS is a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ such that all evaluation functionals are continuous.

In other words, all evaluation functionals

$$\delta_x(f) = \langle f, k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}) \tag{2.19}$$

are continuous if and only if $\mathcal{H}$ is a RKHS, where **nonlinear function $k(\mathbf{x}, \cdot)$ is the representer** (c.f. Example 2.1) that can be also interpreted as the feature map.

The proof of the equivalence between Definition 2.14 and Definition 2.18 can be found in [20].

Corollary 2.1 ([26])**.** *Given an RKHS $\mathcal{H}$, convergence in norm implies point-wise convergence. That is, if*

$$\lim_{n\to\infty} \|f_n - f\|_{\mathcal{H}} = 0 \tag{2.20}$$

then

$$\lim_{n\to\infty} |f_n(\mathbf{x}) - f(\mathbf{x})| = 0, \quad \textit{for all } \mathbf{x} \in \mathcal{X} \tag{2.21}$$

It is readily observed using the fact that for a bounded evaluation functional, then $|f_n(\mathbf{x}) - f(\mathbf{x})| = |\delta_x(f_n) - \delta_x(f)| \leq \|\delta_x\|_{\mathcal{H}}\|f_n - f\|_{\mathcal{H}} \leq M\|f_n - f\|_{\mathcal{H}}$ for some $M \geq 0$. This is an interesting property of RKHS from the viewpoint of optimization: the convergence of functions in the RKHS implies convergence of functions evaluated at each point in the original input space $\mathcal{X}$.

2.3.2 Statistical Learning Theory and Structural Risk Minimization

Corollary 2.1 provides us the possibility and the freedom of searching for the optimal solution in RKHS. In this section, a technique called structural risk minimization [13] is presented to make the concept more tangible.

Risk, Empirical Risk and Family of Functions

Let $\mathbf{x} \in \mathcal{X}$, $\mathbf{y} \in \mathcal{Y}$. Given a loss functional $L : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_0^+$, the *risk* associated with f for any joint probability distribution $\mathbb{P}(\mathbf{x}, y)$ is defined as the expected value of L evaluated at $(\mathbf{x}, y)$:

$$R(f) = \mathbb{E}\left(L(f(\mathbf{x}), y)\right) = \int L\left(f(\mathbf{x}), y\right) d\mathbb{P}\left(\mathbf{x}, y\right) \tag{2.22}$$

A learning algorithm with kernel methods is then designed to search for an optimal $f^* \in \mathcal{F}$, such that

$$f^* = \arg\min_{f \in \mathcal{H}} R(f)$$

In practice, a finite set of training examples $\mathcal{D} = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^N$ are given instead of the distribution $\mathbb{P}(\mathbf{x}, y)$ itself. Hence, the risk $R(f)$ needs to be approximated based on $\mathcal{D}$. This approximation is called the Empirical Risk:

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} L\left(f(\mathbf{x}_i), y_i\right) \tag{2.23}$$

By minimizing $R_{emp}(f)$, we can find the best function $f_{emp}^* \in \mathcal{F}$ for the given training set. However, the ultimate goal of a learning algorithm is to find a function with a good generalization ability, which is indicated by the actual risk $R(f)$ in Eq. (2.22). The gap between $R_{emp}(f)$ and $R(f)$ is then of great interest, where the model complexity plays a key role.

Complexity and VC Dimension

Empirically, the generalization ability can be assessed by the bias/variance trade-off (c.f. Sec. 1.4.1). Intuitively, a more complex $\mathcal{F}$ might lead to a lower bias but a higher variance, since it is prone to over-fitting (c.f. Sec.1.4.2).

An illustration can be found in Fig. 2.4, where model families $\mathcal{F}_1 \cdots \mathcal{F}_n$ are sorted in an ascending order with respect to their complexities with $f_i^* \in \mathcal{F}_i$ denoting the optimal point in each $\mathcal{F}_i$, for $i = 1 \cdots n$. The point f^* indicates the global optimum. We can see that due to a higher complexity, $\mathcal{F}_n$ has a larger search space that includes the global optimum, which implies the possibility of finding f^* by using an efficient searching algorithm. On the contrary, by employing the model family $\mathcal{F}_1$, one simply cannot find the global optimal f^* no matter what searching algorithm one uses, which indicates a larger bias compared to $\mathcal{F}_n$. On the other hand, by using the same

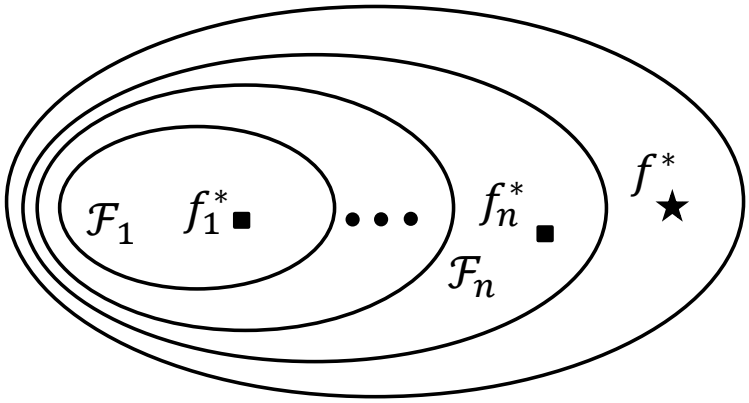

Figure 2.4: The black squares represent the optimal point f_i^* that one can find in the corresponding set $\mathcal{F}_i$, for $i = 1, \cdots, n$. The starred point f^* is the global optimal function that minimizes R. Typically, larger search space $\mathcal{F}_i$ leads to lower bias but higher variance.

searching scheme, a higher variance (w.r.t. different training samples) is typically obtained for $\mathcal{F}_n$ due to the larger search space.

One complexity measure is the *Vapnik-Chervonenkis (VC) dimension* [13], denoted by h. For example, $h_1 < h^* < h_n$ holds for $\mathcal{F}_1 \subset \mathcal{F}^* \subset \mathcal{F}_n$ in Fig. 2.4. Loosely speaking, the VC dimension is the cardinality of the largest subset of $\mathcal{X}$ for which $\mathcal{F}$ can guarantee a zero training error.

To formally define the VC dimension, we first introduce a property of $\mathcal{F}$ called "shattering" [13, 27].

Definition 2.19. A family of functions $\mathcal{F}$ *shatters* a set of data points $\mathcal{S}$ if and only if for every dichotomy [1] of $\mathcal{S}$, there exists some function in $\mathcal{F}$ consistent with this dichotomy.

For instance, a linear classifier in a d dimensional feature space can shatter $d + 1$ data points, which is illustrated in Fig. 2.5. Moreover, Fig. 2.6 shows an example of a dataset consisting of four points that cannot be shattered by a linear classifier in the two dimensional feature space.

Definition 2.20. The Vapnik-Chervonenkis dimension of $\mathcal{F}$ is the maximum number of data points (training data) that $\mathcal{F}$ can shatter.

Note that the definition of the VC dimension h only requires the existence of a set with cardinality h that can be shattered by $\mathcal{F}$. Generally speaking, a family of functions $\mathcal{F}$ with VC dimension h cannot shatter all sets of h points. A demonstration can be found in Fig. 2.7, where a three-point dataset that cannot be shattered by the family of linear classifiers is constructed. However, according to the definition, the VC dimension of $\mathcal{F}$ is still $h = d + 1 = 3$, since one can indeed find a three-point dataset that can be shattered by $\mathcal{F}$ (for instance, Fig. 2.5).

[1] A dichotomy of a set $\mathcal{S}$ is a partition of $\mathcal{S}$ into two disjoint subsets.

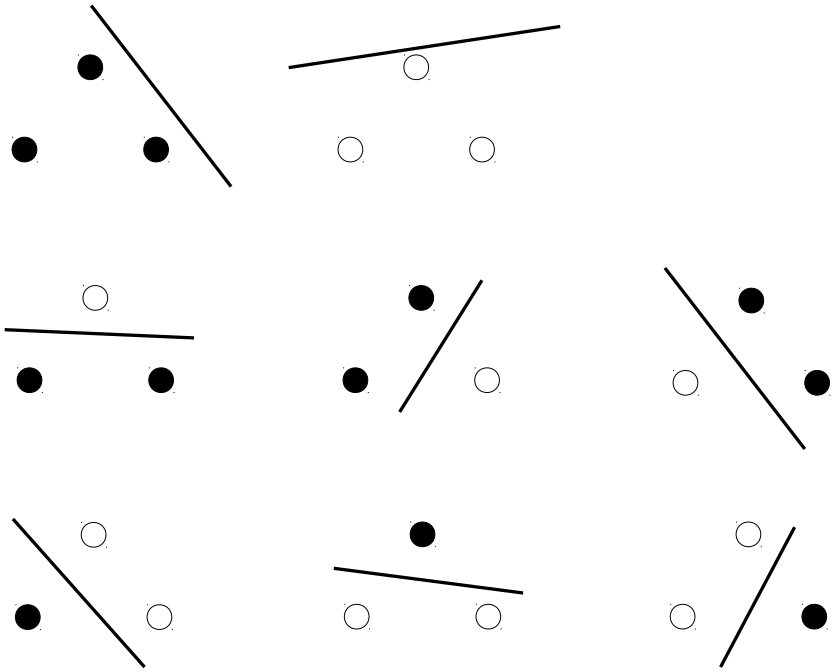

Figure 2.5: This figure illustrates how the family of linear classifiers $\mathcal{F}$ shatters a three-point dataset in a two dimensional space. The face color of the data points (circles) indicates the partition of the set. When a linear function $f \in \mathcal{F}$ is "consistent" with a partition, it means that f can separate the two groups correctly.

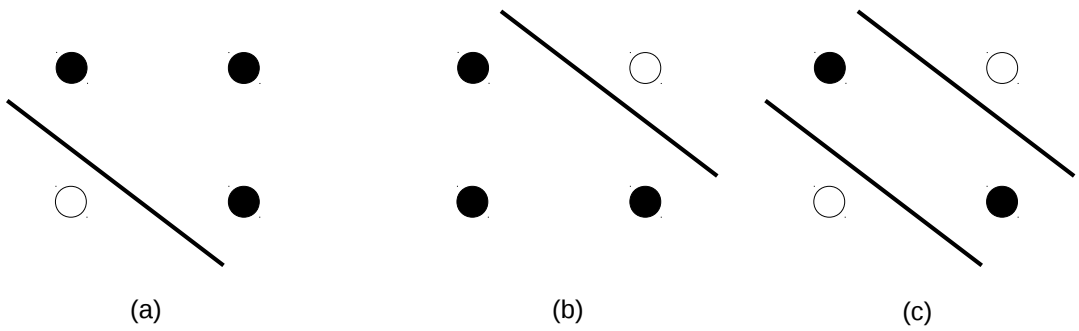

Figure 2.6: The setup in this figure is the same as in Fig. 2.5, but with a four-point dataset instead. In cases (a) and (b), there exists a linear classifier that is consistent with the given partition. However, such a classifier does not exist for (c).

VC Bound and Structural Risk Minimization

A classic result based on VC dimension states that with probability at least $1 - \delta$, every $f \in \mathcal{F}$ satisfies

$$R(f) < R_{emp}(f) + \sqrt{\frac{h\left(\ln \frac{2N}{h} + 1\right) + \ln \frac{4}{\delta}}{N}} \tag{2.24}$$

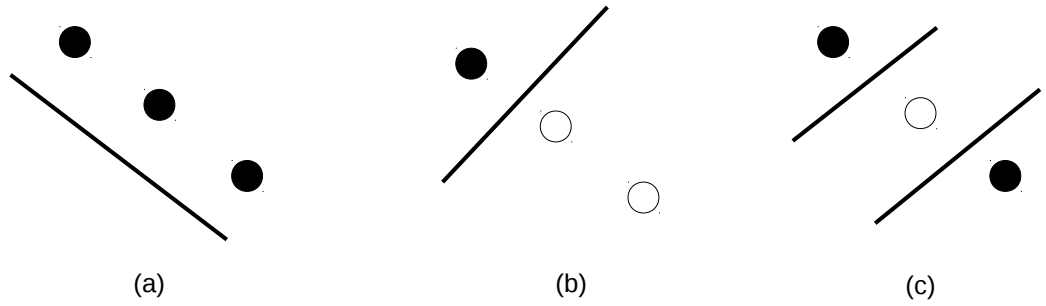

Figure 2.7: This figure shows an example of a three-point dataset that cannot be shattered by the family of linear classifier $\mathcal{F}$. But this existence does not change the fact that the VC dimension of $\mathcal{F}$ is $h = d + 1 = 3$.

where h is the VC dimension of $\mathcal{F}$. This is illustrated in Fig. 2.8.

Therefore, the idea of Structural Risk Minimization (SRM) is to find the simplest function that minimizes the empirical risk function on the finite training set, i.e.,

$$R_{struct}(f) = \frac{1}{N}\sum_{i=1}^{N} L\left(f(\mathbf{x}_i), y_i\right) + \lambda\Omega(f) \tag{2.25}$$

where the penalty term $\lambda\Omega(f)$ is a smooth function that measures the complexity of the function f with λ a positive number controlling the trade-off between bias and variance.

2.3.3 Representer Theorem

Theorem 2.7 (Representer Theorem [9]). *Denoted by $\Omega : [0, \infty) \to \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a set, and by L an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the regularized risk*

$$L\left((f(\mathbf{x}_1), y_1) \cdots (f(\mathbf{x}_N), y_N)\right) + \Omega(\|f\|_{\mathcal{H}})$$

admits a representation of the form

$$f = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i) \tag{2.26}$$

As a result, given a regularized cost function, each minimizer $f(\mathbf{x})$ is then parameterized by the weighting coefficients of all training data vectors. An equivalent formulation, called the Learning Subspace Property (LSP), is given in [11]. LSP states that the subspace property $f \in \text{span}\left(\varphi(\mathbf{x}_1), \cdots, \varphi(\mathbf{x}_N)\right)$ is a sufficient and necessary condition for applying kernel models.

The Representer Theorem motivates and verifies the validity of kernel models. However, the new feature space $\mathcal{H}$ typically has very high dimensionality. In fact,

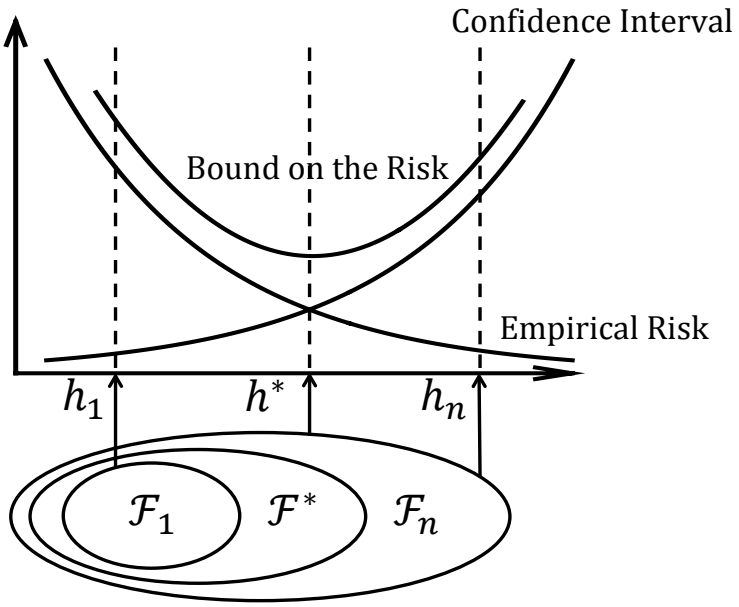

Figure 2.8: Figure 4.2 in [13]: The bound on the risk is the sum of the empirical risk and the confidence interval. The empirical risk decreases with the index of the element of the structure, while the confidence interval increases. The smallest bound of the risk is achieved on some appropriate element of the structure.

$dim(\mathcal{H})$ is allowed to be infinite. In such cases, the explicit expression for f is not practical, or even possible.

2.3.4 Kernel Tricks

The "kernel trick" is a technique applied to resolve the high dimensional problem of $\mathcal{H}$. Briefly speaking, when we formulate a problem using kernel models, data vector $\varphi(\mathbf{x})$'s always show up in the form of inner products with another vector (or itself), rather than appearing alone. Since the inner products can be computed using the predefined kernel function, explicit computations in $\mathcal{H}$ are thus never needed. We illustrate this method with an example.

Example 2.3 (The kernel trick)**.** *Given training pairs $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathbb{R}$ and a predefined kernel function $k(\mathbf{x}, \tilde{\mathbf{x}}) = \varphi(\mathbf{x})^T \varphi(\tilde{\mathbf{x}})$ for any $\tilde{\mathbf{x}}, \mathbf{x} \in \mathcal{X}$, where $\varphi : \mathcal{X} \to \mathcal{H}$. We are interested in finding a vector $\mathbf{w} \in \mathcal{H}$, such that for any unseen data $\varphi(\mathbf{x})$ associated with an output y, $\mathbf{w}^T \varphi(\mathbf{x}) \approx y$. We formulate this problem as a linear regression:*

$$\mathbf{y} = \mathbf{\Phi}^T \mathbf{w} \tag{2.27}$$

where $\mathbf{\Phi} \in \mathbb{R}^{p \times N}$ contains the training data $\varphi(\mathbf{x}_i) \in \mathcal{H}$ on its column, for $N, p \in \mathbb{N}^+$, $p = dim(\mathcal{H})$ and $\mathbf{y} \in \mathbb{R}^N$. The solution to Eq. (2.27) in the least-squares sense is:

$$\mathbf{w}^* = (\mathbf{\Phi}\mathbf{\Phi}^T)^+ \mathbf{\Phi}\mathbf{y} \tag{2.28}$$

where $^+$ *denotes the pseudoinverse of a matrix, which implies that the solution is restricted in the column space of* $\mathbf{\Phi}$*, i.e.*

$$\mathbf{w}^* = \mathbf{\Phi}\boldsymbol{\alpha} \tag{2.29}$$

for some $\boldsymbol{\alpha} \in \mathbb{R}^N$*.*

However, when $dim(\mathcal{H})$ *is large, such as* $p = +\infty$ *in the RBF kernel, computations in Eq.* (2.28) *are prohibitive. In particular, the matrix* $\mathbf{\Phi}\mathbf{\Phi}^T \in \mathbb{R}^{\infty\times\infty}$ *can not be constructed. One employs the kernel trick to resolve such issue.*

By combining Eq. (2.27) *and Eq.* (2.29)*, we have* $\mathbf{y} = \mathbf{\Phi}^T\mathbf{w} = \mathbf{\Phi}^T\mathbf{\Phi}\boldsymbol{\alpha}$ *and hence* $\boldsymbol{\alpha}^* = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{y}$*. Note that all elements in the matrix* $\mathbf{\Phi}^T\mathbf{\Phi}$ *can be computed using the kernel function* $(\mathbf{\Phi}^T\mathbf{\Phi})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ *in the low dimensional space* $\mathcal{X}$*, for* $1 \le i, j \le N$*. That means we do not have to deal with the intrinsic dimension* p*.*

For an unseen data $\tilde{\mathbf{x}}$*, its label* y *is then estimated as*

$$\hat{y} = \mathbf{w}^T\varphi(\tilde{\mathbf{x}}) = \boldsymbol{\alpha}^T\mathbf{\Phi}^T\varphi(\tilde{\mathbf{x}}) = \alpha^T \begin{bmatrix} k(\mathbf{x}_1, \tilde{\mathbf{x}}) \\ \vdots \\ k(\mathbf{x}_N, \tilde{\mathbf{x}}) \end{bmatrix}.$$

The example has shown a simple application of the kernel trick. By using the kernel trick, heavy computations result from the nonlinear mapping $\varphi(\cdot)$ can always be replaced by the kernel function $k(\cdot, \cdot)$, which is operating on the original low dimensional input space.

Example 2.4 (Large data size)**.** *Example 2.3 shows how the kernel trick converts the construction of* $\mathbf{\Phi}\mathbf{\Phi}^T \in \mathbb{R}^{p\times p}$ *into the evaluation of kernel functions* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$*, so that the computational complexity is reduced from* $\mathcal{O}(p^3)$ *for* $\left(\mathbf{\Phi}\mathbf{\Phi}^T\right)^+$ *to* $\mathcal{O}(N^3)$ *for* $\left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}$*. However, the cubic growth with respect to the training size* N *can become impractical for large scale learning problem. Techniques designed to conquer this problem are called kernel approximation, where a low rank structure of the matrix* $\mathbf{\Phi}^T\mathbf{\Phi}$ *is explored. It is presented in details in later chapters.*

2.4 Conclusion

In kernel techniques, the original input space $\mathcal{X}$ is mapped to a high dimensional feature space $\mathcal{H}$, called the Reproducing Kernel Hilbert Space (RKHS), where $dim(\mathcal{H})$ is allowed to be infinite (c.f. Eq. (2.18)). The take-home message of this section is summarized as follows:

- Why Hilbert Space:
  - $\mathcal{H}$ is an inner product space. $\Rightarrow$ The notion of **orthogonality** and **projections** is well defined. **Convergence** is also well defined by the induced norm.

  - $\mathcal{H}$ is **complete**. $\Rightarrow$ Convergence of every Cauchy sequence guarantees convergence of certain greedy algorithms.
  - $\mathcal{H}$ is **assumed** to be **separable** (i.e. it has a countable orthonormal basis). $\Rightarrow$ It allows us to apply finite dimensional linear algebra to (possibly) infinite dimensional inner product space.

- Why RKHS:
  - The Representer theorem for the structural risk minimization.
  - $\mathcal{H}$ is a separable Hilbert Space with the reproducing property (RKHS). $\Leftrightarrow$ Every evaluation functional is continuous. It implies pairwise convergence, i.e. if $f, g \in \mathcal{H}$ are close to each other, then $f(\mathbf{x})$ and $g(\mathbf{x})$ are close for all $\mathbf{x} \in \mathcal{X}$, which is a desirable property for optimization.

- Relation between an inner product and RKHS:
  - An inner product is a positive definite function.
  - Positive Definite Function $\Leftrightarrow$ Reproducing Kernel $\Leftrightarrow$ RKHS.
  - RKHS is unique, but feature spaces associated with the same kernel function is not. Nevertheless, they are all isometrically isomorphic.

- How to find valid kernel functions: from predefined kernels and their combinations.

Chapter 3

Kernel Methods in Practice

It is a very common understanding that kernel methods are related to Support Vector Machines (SVM). Indeed, the primal-dual derivation of SVM (c.f. Eq. (1.3)) naturally leads to the construction of kernel functions and kernel tricks. However, kernel methods are essentially feature transformation techniques that can be employed by most linear learning models. In this chapter, we establish an explicit transformation from linear models to their kernel counterparts using a subspace expression.

We divide kernel methods into three steps: 1) kernel model selection; 2) kernel approximation; and 3) classification training. Fig. 3.1 shows how this chapter is organized.

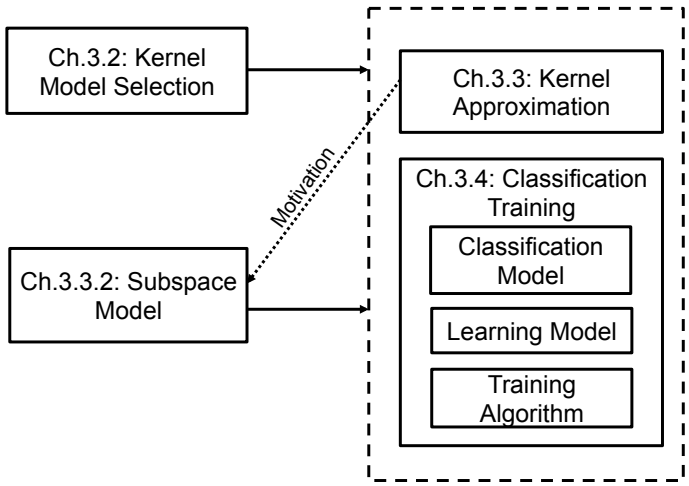

Figure 3.1: The structure of this chapter.

Briefly speaking, kernel model selection determines the kernel type (e.g. RBF, Polynormial, etc.) and kernel parameter (e.g. σ, d, etc.) before the training. It is usually considered a tuning process for hyperparameters. Kernel approximation aims at reducing the model complexity by subset sampling and a subspace transformation. Subset sampling restricts the representation of the classifier to the subspace spanned by a subset of the training set to reduce the computational complexity. RKHS subspace transformation refers to a nonlinear transformation on the selected

subset. Generally, this step has the functionality of further reducing the dimensionality while preserving the data structure. One of the most popular techniques is the PCA whitening in the RKHS. Finally, given the classification model, the learning objective and the training algorithm, a classifier is produced based on the output of the kernel approximation.

3.1 Notations

Given a non-empty input set $\mathcal{X}$, which could be a string of texts, a collection of websites, medical records, etc. Let $\mathcal{F}$ be the RKHS associated with a kernel function k. Denote the mapping $\varphi : \mathcal{X} \to \mathcal{F}$ and the feature vectors $\boldsymbol{\varphi} := \varphi(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$. Without loss of generality, we assume that $\mathbf{x}$'s are vectors with numerical values in this chapter.

In a classification setup, given a training set $\mathcal{D} = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{N}$, $N \in \mathbb{N}$ and a subset $\mathcal{M} \subseteq \mathcal{D}$, the index set is denoted as $\mathcal{I}_{\mathcal{M}} = \left\{i : (\mathbf{x}_i, y_i) \in \mathcal{M}\right\}$. Moreover, let $\mathbf{X}_{\mathcal{M}} = \left[\mathbf{x}_i\right]_{i \in \mathcal{I}_{\mathcal{M}}}$ be a matrix with columns $\mathbf{x}_i$'s and $\boldsymbol{\Phi}_{\mathcal{M}} = \left[\boldsymbol{\varphi}_i\right]_{i \in \mathcal{I}_{\mathcal{M}}}$ be a matrix with columns $\boldsymbol{\varphi}_i$'s. Given a kernel function k, for $\mathcal{M}, \mathcal{N} \subseteq \mathcal{D}$, denote $\mathbf{K}_{\mathcal{M}\mathcal{N}} = k(\mathbf{X}_{\mathcal{M}}, \mathbf{X}_{\mathcal{N}}) = \left[k\left(\mathbf{x}_i, \mathbf{x}_j\right)\right]_{i \in \mathcal{I}_{\mathcal{M}}, j \in \mathcal{I}_{\mathcal{N}}} = \boldsymbol{\Phi}_{\mathcal{M}}^T \boldsymbol{\Phi}_{\mathcal{N}}$ and $\mathbf{K}_{\mathcal{M}} = \boldsymbol{\Phi}_{\mathcal{M}}^T \boldsymbol{\Phi}_{\mathcal{M}}$. In particular, we write $\mathbf{K} = \boldsymbol{\Phi}_{\mathcal{D}}^T \boldsymbol{\Phi}_{\mathcal{D}}$. Furthermore, when the size of any matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ is being emphasized, it is denoted as $\mathbf{M}_{n \times m}$.

Note that in general, for any random vector $\mathbf{a}$, the notation $\mathbf{a}_i$ is reserved for the realization of $\mathbf{a}$ associated with sample number i drawn from the multivariate distribution that generates $\mathbf{a}$.

For the sake of convenience, notations are summarized in Table 3.1.

3.2 Kernel Model Selection

To design learning machines in the RKHS $\mathcal{F}$, we need to construct the space $\mathcal{F}$ first. Theorem 2.5 implies that a (unique) RKHS is generated by a valid kernel function, which can be found in Sec. 2.2.2, where a list of commonly used kernel functions and their valid combinations is provided. Thus, the first step is to determine the RKHS by choosing a valid kernel function.

The choice of the kernel function is twofold: 1) the kernel type and 2) the kernel parameters. The kernel type refers to the parameterization for the inner product in the RKHS. For example, if the kernel type "RBF" is selected (c.f. Eq. (2.12)), an inner product is parameterized using: $\langle \varphi(\mathbf{x}), \varphi(\tilde{\mathbf{x}}) \rangle_{\mathcal{F}} = k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\left(-\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|_2^2}{2\sigma^2}\right)$, for $\sigma \in \mathbb{R}^+$. The *kernel parameter* in this case is then σ, also known as a hyperparameter, which can be determined using techniques such as grid search and cross-validation.

Symbol	Description
$\mathcal{X}$	Non-empty input set.
$\mathbf{x} \in \mathcal{X}$	Input multivariate random variable.
$\mathcal{Y}$	Label set.
$y \in \mathcal{Y}$	Random variable that represents the label information.
$\mathbf{x}_i \in \mathcal{X}$	Data vector sampled from $\mathcal{X}$ associated with sample index i.
$y_i \in \mathcal{Y}$	Label associated with data sample $\mathbf{x}_i$.
N	Training size.
C	Number of classes.
$\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$	Training set.
$\mathcal{I}_\mathcal{M} = \{i : \ (\mathbf{x}_i, y_i) \in \mathcal{M}, \ \mathcal{M} \subseteq \mathcal{D}\}$	Index set associated with subset $\mathcal{M}$.
$[\mathbf{z}_i]_{i \in \mathcal{I}} = [\mathbf{z}_{i_1} \cdots \mathbf{z}_{i_m}]$	A matrix that contains vector $\mathbf{z}_{i_j}$ on its j^{th} column, where $\mathbf{z}_{i_j}$ is from any vector space, and $i_j \in \mathcal{I}$ for index set $\mathcal{I}$, $1 \leq j \leq m = \lvert\mathcal{I}\rvert$ (the cardinality of $\mathcal{I}$).
$\mathbf{X}_\mathcal{M} = [\mathbf{x}_i]_{i \in \mathcal{I}_\mathcal{M}}$	Data matrix.
$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	Kernel function.
$\mathcal{F}$	The RKHS associated with k.
$\varphi : \mathcal{X} \to \mathcal{F}$	Nonlinear map such that $\varphi(\cdot) = k(\mathbf{x}, \cdot)$
$\boldsymbol{\varphi} = \varphi(\mathbf{x}) \in \mathcal{F}$	Random vectors in $\mathcal{F}$.
$\boldsymbol{\varphi}_i = \varphi(\mathbf{x}_i)$	Data vector sampled from $\mathcal{F}$.
$\boldsymbol{\Phi}_\mathcal{M} = [\boldsymbol{\varphi}_i]_{i \in \mathcal{I}_\mathcal{M}}$	Data matrix in RKHS.
$k(\mathbf{X}_\mathcal{M}, \mathbf{X}_\mathcal{N})$	A matrix, where the $(i,j)^{th}$ entry of the matrix is $k(\mathbf{X}_\mathcal{M}(:,i), \mathbf{X}_\mathcal{N}(:,j))$, for all $i \in \mathcal{I}_\mathcal{M}$ and $j \in \mathcal{I}_\mathcal{N}$.
$\mathbf{K} = \boldsymbol{\Phi}_\mathcal{D}^T \boldsymbol{\Phi}_\mathcal{D} = k(\mathbf{X}_\mathcal{D}, \mathbf{X}_\mathcal{D})$	The Gram matrix.
$\mathbf{K}_{\mathcal{M}\mathcal{N}} = \boldsymbol{\Phi}_\mathcal{M}^T \boldsymbol{\Phi}_\mathcal{N} = k(\mathbf{X}_\mathcal{M}, \mathbf{X}_\mathcal{N})$	Kernel matrix computed on two subsets.
$\mathbf{K}_\mathcal{M} = \boldsymbol{\Phi}_\mathcal{M}^T \boldsymbol{\Phi}_\mathcal{M} = k(\mathbf{X}_\mathcal{M}, \mathbf{X}_\mathcal{M})$	
$\mathbf{I}_{k \times k} \in \mathbb{R}^{k \times k}$	An $k \times k$ identity matrix.

Table 3.1: Notations used in this chapter.

3.2.1 Cross-Validation

Briefly speaking, *cross-validation* [28, 29, 30] divides training data into two disjoint subsets: the training set and the validation set. Given the search space for hyperparameters, the machine learning method is trained on the training set and evaluated on the validation set. The "best" value is then selected with respect to the best performance evaluated on the validation set. Note that any objective function can be used as the criterion for finding hyperparameters.

In *n-fold cross-validation*, the whole training dataset is divided into n disjoint subsets, amongst which, one subset is used for validation and the remainder for training. One shall repeat such procedure until all data have been used for both training and validation. A special n-fold cross-validation technique is called the *Leave-One-Out (LOO) validation*, where $n = N$ the training sample size. The trained algorithm is tested on the i^{th} training data point for all $i = 1 \cdots N$, where data points from $\mathcal{I}_{\mathcal{D}} \setminus i$ are used for training.

3.2.2 Multiple-Kernel (MK) Model

In kernel methods, a positive definite kernel function is selected to construct the high dimensional feature space. The design of kernel functions is a subject for itself. Nevertheless, from Sec. 2.2.2, we know that we can generate "new" kernel functions by applying certain combinations of different existing kernels, which might have a better potential to provide a more flexible and robust solution. In the literature, this is called the Multiple-Kernel (MK) models.

Briefly speaking, given a (typically finite and countable) set of kernel functions, one constructs a new valid kernel function k_{MK} by applying a linear (such as weighted sum) or a nonlinear (such as multiplication) function to the members from the set to gain better flexibility, i.e.

$$k_{MK} = h(k_1, \cdots, k_M; \Theta)$$

where $h : \mathcal{K} \mapsto k_{MK}$ is parameterized by a set of parameters Θ. When Θ is unknown, the estimation of Θ is a learning process itself. As shown in Sec. 1.2, there are three key components in a learning process: the learning model, the learning objective, and the searching algorithm. A comprehensive review can be found in [31], where various techniques are elaborated and compared for each learning component. Generally speaking, by selecting the MK model h, the learning objectives for estimating Θ are categorized by [31] into i) similarity-based functions, ii) structural risk function and iii) Bayesian function. Moreover, according to its strategy, the learning objective can be further classified into 1) one-step methods, where Θ and the parameters for the predictor are estimated in one run; and 2) two-step methods, where the searching is alternating between finding the optimal MK parameters Θ and the optimal solution for the predictor. The learning objective is usually formulated as a Semidefinite Programming (SDP) or a Quadratically Constrained Quadratic Program (QCQP), where standard algorithms for solving convex optimization problems can be applied, such as the interior point method.

Example 3.1 (MK generated from weighted addition)**.** *Let $\mathcal{K} = \{k_1, \cdots, k_M\}$ be a set of kernel functions, $M \in \mathbb{N}^+$. A MK model can be selected as (c.f. Sec. 2.2.2):*

$$k_{MK} = \sum_{i=1}^{M} a_i k_i \tag{3.1}$$

where $a_i \in \{0\} \cup \mathcal{R}^+$ and $\sum_{i=1}^{M} a_i = 1$. Hence, instead of using one single kernel from $\mathcal{K}$, we construct a RKHS by a linear combination of all kernel functions in the set, where α_i's are unknown parameters that need to be estimated. Given Eq. (3.1)*, the estimation of α_i's depends on the learning objective and the searching algorithm. For example, given training vectors $\mathbf{X}_{\mathcal{D}}$ one of the most straightforward objective function is:*

$$\begin{aligned} \underset{\alpha_1,\cdots,\alpha_M}{\textit{minimize:}} \quad & \left\| \mathbf{K}^* - \sum_{i=1}^{M} \alpha_i \mathbf{K}_i \right\|_F \\ \textit{subject to:} \quad & \alpha_i \geq 0, \ \sum_{i=1}^{M} \alpha_i = 1 \end{aligned} \tag{3.2}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbf{K}_i = k_i(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{D}})$. The matrix $\mathbf{K}^*$ is the optimal kernel matrix corresponding to the classification task. For example, the optimal kernel matrix for a binary classification problem is:

$$\mathbf{K}^*_{binary} = \mathbf{y}\mathbf{y}^T = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix}, \quad \textit{where } y_i \in \{-1, +1\}.$$

Convex optimization techniques can be applied accordingly.

3.3 Kernel Approximation

3.3.1 Motivation

As shown in Sec. 2.3.4, when it comes to kernel methods, we are always encouraged to take advantage of the kernel trick. Indeed, the kernel trick is the portal that connects the high dimensional RKHS $\mathcal{F}$ (or its isometrically isomorphic equivalences) and our computational world with a limited computational power. In practice, it means that we only use the kernel matrix $\mathbf{K}$ to describe data in RKHS through the kernel trick. More precisely, given a kernel function k, the kernel matrix $\mathbf{K}$ can be computed using $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, for $1 < i, j \leq N$. Once $\mathbf{K}$ is computed, the entire data structure is characterized by $\mathbf{K}$ and the original training data can be left aside for the classification training.

However, by completely relying on the kernel matrix, one might suffer from the following drawbacks:

- High computational complexity and storage requirement. With a growing sampling size N, the demand for computations and storage for a $N \times N$ kernel matrix $\mathbf{K}$ becomes prohibitive.

- Low flexibility of the learning algorithm, i.e., one always needs to reformulate linear techniques into their "kernelized" version, where only inner products of data vectors are involved, so that training data can be replaced by the kernel matrix. This turns out to be inconvenient in some scenarios.

- Overfitting problem. The high (possibly infinite) dimensional space $\mathcal{F}$ implies high model complexity. Moreover, the Representer Theorem 2.7 states that the intrinsic data structure is determined by the subspace spanned by training vectors in $\mathcal{F}$. This highly data dependent learning model with high model complexity typically leads to overfitting.

To resolve the above issues, kernel approximation techniques are introduced, where the kernel matrix is assumed to have a low rank structure. A $N \times r$ matrix $\mathbf{G}$ is then constructed with $r \ll N$, such that $\mathbf{K} \approx \mathbf{G}\mathbf{G}^T$.

In many kernel techniques, such as Least-Square SVM (LS-SVM), Gaussian Processes (GP), Kernel Principal Component Analysis (KPCA), etc, given ρ the regularization parameter [32] and $\mathbf{I}$ the identity matrix, one needs to invert the regularized kernel matrix $(\mathbf{K} + \rho\mathbf{I})$, which results in a computational cost of order $\mathcal{O}(N^3)$. By constructing the matrix $\mathbf{G}$, there are two ways of utilizing this approximation to reduce the computational complexity:

i. Matrix inversion by the Woodbury identity [33]:

$$(\mathbf{K} + \rho\mathbf{I})^{-1} \approx \frac{1}{\rho} - \frac{1}{\rho^2}\mathbf{G}\underbrace{\left(\mathbf{I} + \frac{1}{\rho}\mathbf{G}^T\mathbf{G}\right)^{-1}}_{\in\mathbb{R}^{r\times r}}\mathbf{G}^T \tag{3.3}$$

ii. Using the explicit feature vector:

From $\mathbf{K} \approx \mathbf{G}\mathbf{G}^T$, one can construct an explicit feature vector:

$$\boldsymbol{\beta}_i = \mathbf{G}(i,:)^T \in \mathbb{R}^r \tag{3.4}$$

where $\mathbf{G}(i,:)$ denotes the i^{th} row of matrix $\mathbf{G}$, and linear techniques can be applied in the space $\mathbb{R}^r$ consequently.

In both cases, the complexity is reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(r^3 + rN)$.

3.3.2 Subspace Model for Kernel Approximation

Aiming at potential problems such as high computational complexity and overfitting, an explicit subspace model is suggested, where feature vectors are assumed to be a composition of a *"noise-free"* part and the *noise* vector. The noise-free vector takes values in a *deterministic subspace*, denoted by $\mathcal{V} \subset \mathcal{F}$, and the noise vector lies in the whole space $\mathcal{F}$.

More precisely, given a matrix $\mathbf{U} \in \mathbb{R}^{q \times r}$ with orthonormal columns, where $q = dim(\mathcal{F})$ and $r < q$. Let $\boldsymbol{\psi} \in \mathcal{V}$ be the noise-free coordinates and $\zeta \in \mathcal{F}$ the zero-mean random noise vector with a finite covariance. For a given data vector $\boldsymbol{\varphi}$, the underlying model assumption is as follows:

$$\begin{aligned} \boldsymbol{\varphi} &= \mathbf{U}\boldsymbol{\psi} + \zeta && (3.5)\\ &= \mathbf{U}\boldsymbol{\psi} + \mathbf{U}\mathbf{U}^T\zeta + \mathbf{U}_\perp\mathbf{U}_\perp^T\zeta && (3.6)\\ &= \mathbf{U}\underbrace{\left(\boldsymbol{\psi} + \mathbf{U}^T\zeta\right)}_{\boldsymbol{\beta}} + \underbrace{\mathbf{U}_\perp\mathbf{U}_\perp^T\zeta}_{\mathbf{e}} \\ &= \mathbf{U}\boldsymbol{\beta} + \mathbf{e} && (3.7) \end{aligned}$$

where $\boldsymbol{\beta}$ is the random vector containing coordinates in the subspace spanned by $\mathbf{U}$ and the residual $\mathbf{e} \in \mathcal{F} \setminus \mathcal{V}$, i.e. $\mathbf{U} \perp \mathbf{e}$.

In the typically high dimensional RKHS, a subspace model makes perfect sense, since it implies the underlying assumption that the data distribution is not completely arbitrary in the ambient space $\mathcal{F}$. Instead, data are clustering around a low dimensional subspace.

Low Rank Approximation

Given assumption Eq. (3.7) and $\mathbf{U} \in \mathbb{R}^{q \times r}$ with $r \ll N \ll q$, $\mathbf{K}$ can be decomposed as follows:

$$\begin{aligned} \mathbf{K} &= \Big[\mathbf{U}\boldsymbol{\beta}_1 + \mathbf{e}_1, \ \cdots, \ \mathbf{U}\boldsymbol{\beta}_N + \mathbf{e}_N\Big]^T \Big[\mathbf{U}\boldsymbol{\beta}_1 + \mathbf{e}_1, \ \cdots, \ \mathbf{U}\boldsymbol{\beta}_N + \mathbf{e}_N\Big] \\ &= \Big[\mathbf{U}\mathbf{B} + \mathbf{E}\Big]^T \Big[\mathbf{U}\mathbf{B} + \mathbf{E}\Big] \\ &= \mathbf{B}^T\mathbf{B} + \mathbf{E}^T\mathbf{E} \\ &= \boldsymbol{\Phi}_{\mathcal{D}}^T\mathbf{U}\mathbf{U}^T\boldsymbol{\Phi}_{\mathcal{D}} + \mathbf{E}^T\mathbf{E} && (3.8) \end{aligned}$$

where $\mathbf{B} = \Big[\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_N\Big]$ and $\mathbf{E} = \Big[\mathbf{e}_1 \cdots \mathbf{e}_N\Big]$. Given the Frobenius norm $\|\cdot\|_F$, if $\|\mathbf{E}^T\mathbf{E}\|_F \leq B$, for $B \in \mathbb{R}$ and $B < +\infty$, we have:

$$\|\mathbf{K} - \boldsymbol{\Phi}_{\mathcal{D}}^T\mathbf{U}\mathbf{U}^T\boldsymbol{\Phi}_{\mathcal{D}}\|_F \leq B \tag{3.9}$$

i.e. $\mathbf{K}$ can be approximated using:

$$\mathbf{K} \approx \boldsymbol{\Phi}_{\mathcal{D}}^T\mathbf{U}\mathbf{U}^T\boldsymbol{\Phi}_{\mathcal{D}} \tag{3.10}$$

up to an error bounded by B. Hence, Eq. (1) can be applied to the learning model, where

$$\mathbf{G} = \boldsymbol{\Phi}_{\mathcal{D}}^T\mathbf{U}. \tag{3.11}$$

Explicit Feature Map

Given Eq. (2) and Eq. (3.11), we know that for any $\varphi(\mathbf{x}) \in \mathcal{F}$, an explicit feature map β can be constructed as:

$$\beta(\mathbf{x}) = \mathbf{U}^T\varphi(\mathbf{x}) \in \mathbb{R}^r \tag{3.12}$$

Explicit feature map enables us to employ linear classification techniques in $\mathbb{R}^r$, which leads to a nonlinear boundary in the original input space.

Remark. Note that by using Eq. (3.11) and Eq. (1), we still work in a subspace of the RKHS, whereas Eq. (3.12) leads us to a high dimensional Euclidean space, where linear techniques are applied. Despite being algebraically equivalent from a kernel approximation point of view, they lead to different classification models and formulations for the classification training. We shall see the illustration of this in Section 3.4.

3.3.3 Subspace Estimation

In practice, the subspace basis matrix $\mathbf{U}$ is unknown and needs to be estimated from training data. This is typically done by finding an index set $\mathcal{I}_\mathcal{G}$ with $\mathcal{G} \subseteq \mathcal{D}$, such that

$$\left\|\mathbf{K} - k(\mathbf{X}_\mathcal{D}, \mathbf{X}_\mathcal{G})k(\mathbf{X}_\mathcal{G}, \mathbf{X}_\mathcal{G})^+ k(\mathbf{X}_\mathcal{G}, \mathbf{X}_\mathcal{D})\right\|_* \tag{3.13}$$

is small, where $\|\cdot\|_*$ is any matrix norm. Given $\mathcal{I}_\mathcal{G}$, to obtain the basis matrix $\mathbf{U}$ with orthonormal columns, Kernel Principal Component Analysis (KPCA) whitening [34, 35, 36] is applied. Hence, the output are the index set $\mathcal{I}_\mathcal{G}$ and a transformation matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{G}|\times r}$, such that $\mathbf{A}^T k(\mathbf{X}_\mathcal{G}, \mathbf{X}_\mathcal{G})\mathbf{A} = \mathbf{I}_{r\times r}$.

There are various kernel approximation techniques existing in the literature, some of which are summarized below.

- Based on Random Sampling

  - Fourier Transform: Bochner's Theorem [37] states that every kernel function k is the inverse Fourier transform of a non-negative measure. In particular, if k is normalized, it is the inverse Fourier transform of a proper probability distribution. This enables the possibility of a random parameterization of k, whose solution can be approximated by Monte Carlo methods [38].

  - Nyström Method and Its Modifications: Another popular class of low-rank approximation algorithms are from the family of Nyström methods [39, 40, 41]. Briefly speaking, they select random columns from kernel matrix $\mathbf{K}$ to construct a matrix $\mathbf{C}$, such that $\mathbf{K} \approx \mathbf{C}\mathbf{W}\mathbf{C}^T$, where $\mathbf{C} \in \mathbb{R}^{N\times d}$ for $d \leq N$ and $\mathbf{W} \in \mathbb{R}^{d\times d}$ is the intersection of the selected columns and the corresponding rows in the kernel matrix $\mathbf{K}$. Different sampling techniques (such as [42]) are proposed and a comparison can be found in [43].

    In [44], the authors have concluded that when there is a large gap in the eigenspectrum of the kernel matrix, Nyström based methods outperform random Fourier features in terms of generalization error bound.

- Based on Subspace Innovation Another family of techniques interpret the kernel matrix approximation as a subspace estimation problem. Given a training sample, a subspace is iteratively constructed by the selection and transformation of the "innovative" subset [45].

  - Incomplete Cholesky Decomposition: One of the most commonly applied techniques is the incomplete Cholesky decomposition [46, 47], where the kernel matrix is approximated by $\mathbf{K} \approx \mathbf{R}(1:d,:)^T\mathbf{R}(1:d,:)$ with $d < N$. This is realized by the Gram-Schmidt process (c.f. 2.1.1) applied on the training data in a sequential manner, where only the pivot vectors are kept to construct $\mathbf{R}$, such that $\mathbf{K} = \mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R} = \mathbf{R}^T\mathbf{R}$. Extensions using side information are proposed in [48], where they take into consideration not only the reconstruction error $\|\mathbf{K}-\mathbf{R}(1:d,:)^T\mathbf{R}(1:d,:)\|$, but also the objective function for the classifier.
  - Greedy Spectral Embedding: In [45], a greedy search algorithm called Greedy Spectral Embedding (GSE) is presented, where they select basis vectors incrementally to minimize an upper bound of the approximation error in an iterative manner. That is, for a new training data, if the projection distance to the constructed subspace is high, it is considered "informative" and hence selected to be a part of the approximation. The resulting selected basis vectors is a subset of the columns in the full kernel matrix $\mathbf{K}$.

- Based on Special Structures

  Some techniques approximate $\mathbf{K}$ by utilizing special structure within the kernel matrix, such as the Memory Efficient Kernel Approximation (MEKA) [49], where dominating behaviors of the diagonal sub-matrices are observed after clustering the data in the original input space. Data vectors are first clustered and basis vectors for the diagonal blocks are estimated, which leads to small values on the off-diagonal blocks.

3.4 Classification Training

3.4.1 Overview

Classification training consists of three essential elements:

1) **Classification model:**

   In kernel methods, the classification model refers to the parameterization of the prediction function $f : \mathcal{X} \to \mathbb{R}$, where $f \in \mathcal{F}$. In particular, for a binary classification problem, the label y for an unseen data vector $\mathbf{x}$ can be estimated as:

$$\hat{y} = \begin{cases} +1, & f(\mathbf{x}) \geq 0 \\ -1, & f(\mathbf{x}) < 0 \end{cases}$$

2) **Learning objective:**

The learning objective specifies optimization criteria for finding a desirable f^* for a specific parameterization. The learning objective usually consists of an objective function to be optimized and some constraints to restrict the search space. The choice of the learning objective determines the bias-variance trade-off via the model complexity.

3) **Training algorithm:**

Given the classification model f and the learning objective, the training algorithm is the approach that one employs for finding the unknown parameters in f by optimizing the objective function in the learning objective with respect to the given constraints. The training algorithm produces the final output of the classification training, which is the classifier f^*.

Example: For an illustrative example, recall the Support Vector Machine (Sec. 1.3):

- **Classification model:** $f = \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}_i$.

- **Learning objective:**

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize:}} \quad & \sum_j \alpha_j - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\varphi}_i^T \boldsymbol{\varphi}_j \\ \text{subject to:} \quad & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \ \forall i \end{aligned} \tag{3.14}$$

- **Training algorithm:** Solve Eq. (3.14) for optimal $\alpha_i's$ using appropriate algorithms, such as the Sequential Minimal Optimization (SMO) algorithm [50].

The output of the classification training is then the parameterization of f: $\alpha_1^*, \cdots, \alpha_N^*$.

SVM is a classic application of the Structural Risk Minimization (SRM) framework presented in Sec. 2.3. Generally speaking, the objective function of SRM has the following form:

$$R(f) = \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i) + \lambda \Omega(\|f\|_{\mathcal{F}}) \tag{3.15}$$

where $\Omega : \ [0, \infty) \to \mathbb{R}$ is a strictly monotonic increasing function and L is an arbitrary loss function. The parameterization of f is given by the Representer Theorem (c.f. Theorem 2.7): $f = \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}_i$ for unknown $\alpha_i \in \mathbb{R}$. Both Ω and L are thus parameterized by α_i's, which then can be solved by a proper learning algorithms.

Classification model		(a)	$f_a = \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}_i \in \mathcal{F}$
		(b)	$f_b = \sum_{i=1}^{N} \alpha_i \mathbf{U}\boldsymbol{\beta}_i \in \mathcal{F}$
		(c)	$f_c = \begin{bmatrix} w_1 & \cdots & w_r \end{bmatrix}^T \in \mathbb{R}^r$
Learning objective	Objective L	(a)	$f_a(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \boldsymbol{\beta}_i^T \boldsymbol{\beta} + \sum_{j=1}^{N} \alpha_j \mathbf{e}_j^T \mathbf{e}$ $= \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x})$
		(b)	$f_b(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \boldsymbol{\beta}_i^T \boldsymbol{\beta}$ $= \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{X}_{\mathcal{G}}) \mathbf{A}\mathbf{A}^T k(\mathbf{X}_{\mathcal{G}}, \mathbf{x})$
		(c)	$f_c(\mathbf{x}) = \sum_{i=1}^{r} w_i \mathbf{v}_i^T \boldsymbol{\beta}$ $= \sum_{i=1}^{r} w_i \mathbf{v}_i^T \mathbf{A}^T k(\mathbf{X}_{\mathcal{G}}, \mathbf{x})$
	Regularizer Ω	(a)	$\Omega(\|f_a\|_{\mathcal{F}}) = \boldsymbol{\alpha}^T \boldsymbol{\Phi}_{\mathcal{D}}^T \boldsymbol{\Phi}_{\mathcal{D}} \boldsymbol{\alpha}$
		(b)	$\Omega(\|f_b\|_{\mathcal{F}}) = \boldsymbol{\alpha}^T \mathbf{B}^T \mathbf{B} \boldsymbol{\alpha}$
		(c)	$\Omega(\|f_c\|_2) = \mathbf{w}^T \mathbf{w}$

Table 3.2: This table shows how different parameterizations of kernel approximation affect the classification training for a given noise-free subspace basis matrix $\mathbf{U}$.

3.4.2 Connection to Kernel Approximation:

Different outputs from the kernel approximation step lead to various classification models. More explicitly, we have three possible outcomes from kernel approximation: (a) full kernel matrix without approximation; (b) low rank approximation; and (c) the explicit feature map, which can be respectively expressed as follows:

$$\mathbf{K} \overset{(a)}{=} \begin{bmatrix} \boldsymbol{\varphi}_1^T \\ \vdots \\ \boldsymbol{\varphi}_N^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\varphi}_1 & \cdots & \boldsymbol{\varphi}_N \end{bmatrix} \tag{3.16}$$

$$\left.\begin{aligned} &\overset{(b)}{\approx} \begin{bmatrix} (\mathbf{U}\mathbf{U}^T\boldsymbol{\varphi}_1)^T \\ \vdots \\ (\mathbf{U}\mathbf{U}^T\boldsymbol{\varphi}_N)^T \end{bmatrix} \begin{bmatrix} \mathbf{U}\mathbf{U}^T\boldsymbol{\varphi}_1 & \cdots & \mathbf{U}\mathbf{U}^T\boldsymbol{\varphi}_N \end{bmatrix} \\ &\overset{(c)}{=} \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_N^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 & \cdots & \boldsymbol{\beta}_N \end{bmatrix} \end{aligned}\right\} \text{Kernel Approximation} \tag{3.17}$$

where $\mathbf{U}\mathbf{U}^T\boldsymbol{\varphi}_i^T$ is the Projection onto subspace $col(\mathbf{U})$ and $\boldsymbol{\beta}_i = \mathbf{U}^T\boldsymbol{\varphi}_i = \mathbf{A}^T k(\boldsymbol{\Phi}_{\mathcal{G}}, \mathbf{x}_i)$ is the explicit feature vector, for $i = 1, \cdots, N$.

As aforementioned, despite of being algebraically equivalent from a kernel approximation point of view, (b) and (c) results in different classification models due to the different spaces they live in. By using (b), the classifier is a vector in $\mathcal{F}$ while being restricted to the column space of $\mathbf{U}$. On the other hand, using (c) implies that the classifier is a linear vector in $\mathbb{R}^r$. In particular, from Eq. (3.15), we know that for a

given classification model f, the learning objective L depends the evaluation of f.

More precisely, given Eq. (3.17) and the subspace model $\varphi(\mathbf{x}) = \mathbf{U}\boldsymbol{\beta} + \mathbf{e}$, where $\mathbf{e}$ represents the residual, various classification models result in different evaluations as follows.

Evaluation of f_a: When using the classification model (a), no kernel approximation is applied. From the Representer Theorem, we have:

$$\begin{aligned} f_a(\mathbf{x}) &= \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}_i^T \varphi(\mathbf{x}) \\ &= \sum_{i=1}^{N} \alpha_i \left(\mathbf{U}\boldsymbol{\beta}_i + \mathbf{e}_i\right)^T \left(\mathbf{U}\boldsymbol{\beta} + \mathbf{e}\right) \\ &= \sum_{i=1}^{N} \alpha_i \boldsymbol{\beta}_i^T \boldsymbol{\beta} + \sum_{j=1}^{N} \alpha_j \mathbf{e}_j^T \mathbf{e} \\ &= \sum_{i=1}^{N} \alpha_i k\left(\mathbf{x}_i, \mathbf{x}\right) \end{aligned} \tag{3.18}$$

Evaluation of f_b: The kernel function is projected onto the subspace spanned by the columns of $\mathbf{U}$ by using classification model (b), which leads to the following expression for the evaluation:

$$\begin{aligned} f_b\left(\mathbf{x}\right) &= \left(\sum_{i=1}^{N} \alpha_i \mathbf{U}\mathbf{U}^T \boldsymbol{\varphi}_i\right)^T \mathbf{U}\mathbf{U}^T \varphi(\mathbf{x}) \\ &= \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}_i^T \boldsymbol{\Phi}_{\mathcal{G}} \mathbf{A}\mathbf{A}^T \boldsymbol{\Phi}_{\mathcal{G}}^T \boldsymbol{\varphi} \\ &= \sum_{i=1}^{N} \alpha_i k\left(\mathbf{x}_i, \mathbf{X}_{\mathcal{G}}\right) \mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{x}\right) \end{aligned} \tag{3.19}$$

where $\boldsymbol{\Phi}_{\mathcal{G}}$ and $\mathbf{A} \in \mathbb{R}^{|\mathcal{G}| \times r}$ are defined in Sec. 3.3.3, where $|\mathcal{G}| \ll N$.

Evaluation of f_c: When using the explicit feature map, for given $\{\mathbf{v}_1, \cdots, \mathbf{v}_r\}$ a set of orthonormal vectors in $\mathbb{R}^r$, we have:

$$\begin{aligned} f_c\left(\mathbf{x}\right) &= \sum_{i=1}^{r} w_i \mathbf{v}_i^T \left(\mathbf{U}^T \varphi(\mathbf{x})\right) \\ &= \sum_{i=1}^{r} w_i \mathbf{v}_i^T \boldsymbol{\beta} \\ &= \sum_{i=1}^{r} w_i \mathbf{v}_i^T \underbrace{\mathbf{A}^T \boldsymbol{\Phi}_{\mathcal{G}}^T \boldsymbol{\varphi}}_{\boldsymbol{\beta}} \\ &= \sum_{i=1}^{r} w_i \mathbf{v}_i^T \mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{x}\right) \end{aligned} \tag{3.20}$$

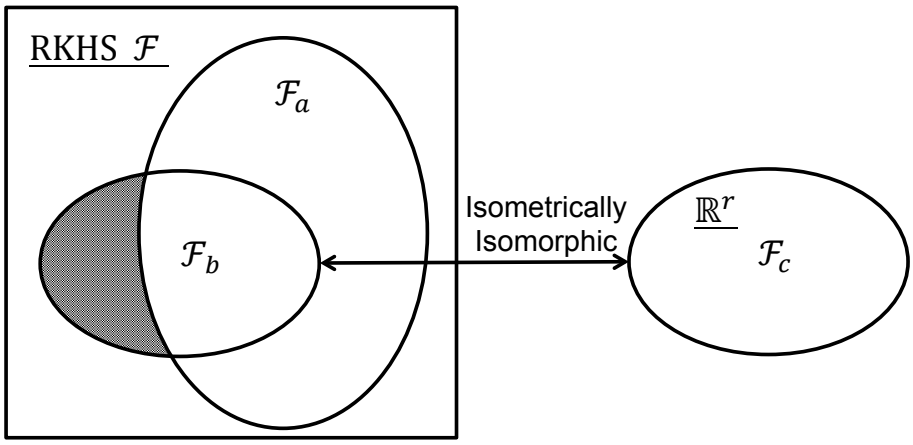

Figure 3.2: $\mathcal{F}_a$, $\mathcal{F}_b$ and $\mathcal{F}_c$ are the search space specified by parameterizations f_a, f_b and f_c, respectively. As we can see that $\mathcal{F}_a, \mathcal{F}_b \subset \mathcal{F}$, whereas $\mathcal{F}_c \subseteq \mathbb{R}^r$. The shadowed part of $\mathcal{F}_b$, which represents the subspace spanned by data that are not a part of the training set, is "invisible" to us. When the basis matrix $\mathbf{U}$ is unknown and needs to be estimated from training data, it can only be derived as a subspace spanned by the training data, but not the shadowed part.

One achieves faster evaluation time by using Eq. (3.20), since $r < N$.

The comparison is summarized in Table 3.2 and Fig. 3.2, where search spaces $\mathcal{F}_a$, $\mathcal{F}_b$ and $\mathcal{F}_c$ correspond to parameterizations f_a, f_b and f_c, respectively. From the analysis in this section, we know that $\mathcal{F}_a = span\,(\boldsymbol{\varphi}_1 \cdots \boldsymbol{\varphi}_N) \subset \mathcal{F}$ and $\mathcal{F}_b = span\,(\mathbf{u}_1 \cdots \mathbf{u}_r) \subset \mathcal{F}$, where $r \ll N$. The shadowed subset indicates the subspace

$$Shadow = span\,(\mathbf{u}_1 \cdots \mathbf{u}_r) \setminus (span\,(\mathbf{u}_1 \cdots \mathbf{u}_r) \cap span\,(\boldsymbol{\varphi}_1 \cdots \boldsymbol{\varphi}_N))$$

which is unknown to us if the subspace $span\,(\mathbf{u}_1 \cdots \mathbf{u}_r)$ is not given but estimated from training data. Given the subspace model in Eq. (3.7), search space $\mathcal{F}_a$ leads to a higher variance compared to $\mathcal{F}_b$ (c.f. Fig. 2.4).

Furthermore, for any given $\mathbf{U} = \begin{bmatrix}\mathbf{u}_1 & \cdots & \mathbf{u}_r\end{bmatrix}$, $\mathcal{F}_b$ and $\mathcal{F}_c$ are isometrically isomorphic (c.f. Eq. (3.17)), which means they are spaces induced by the same kernel function.

3.4.3 Example: LS-SVM

We use the LS-SVM as an example to demonstrate what has been discussed so far. Briefly speaking, LS-SVM finds the separating hyperplane by tuning the marginal hyperplanes, which is done by fitting all training data to the closest marginal hyperplane.

Given training pairs $(\boldsymbol{\varphi}_i, y_i)$ with $y_i \in \{-1, +1\}$ for $i = 1 \cdots N$ and a hyperparameter $\eta \in \mathbb{R}^+$, which controls the trade-off between the size of the soft margin and how many misclassified patterns are allowed within the marginal hyperplanes. Compared

to Eq. (1.2), the *Learning Objective* for LS-SVM is formulated as follows:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize:}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\eta}{2}\sum_{i=1}^{N}\xi_i^2 \\ \text{subject to:} \quad & \xi_i = 1 - \left(\mathbf{w}^T\boldsymbol{\varphi}_i + b\right) y_i \end{aligned} \tag{3.21}$$

or alternatively,

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize:}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\eta}{2}\sum_{i=1}^{N}e_i^2 \\ \text{subject to:} \quad & e_i = y_i - \left(\mathbf{w}^T\boldsymbol{\varphi}_i + b\right) \end{aligned} \tag{3.22}$$

Note that Eq. (3.21) and Eq. (3.22) are equivalent, where Eq. (3.22) is also suitable for regression tasks.

- **Training Algorithm Option 1):** training in the dual:
  - Lagrangian of Eq. (3.22):

$$L(\mathbf{w}, b, e_1, \cdots, e_N, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\eta}{2}\sum_{i=1}^{N}e_i^2 - \sum_{i=1}^{N}\alpha_i\left(\mathbf{w}^T\boldsymbol{\varphi}_i + b + e_i - y_i\right) \tag{3.23}$$

    where α_i's are the Lagrangian Multipliers.
  - KKT conditions:

    Taking derivative with respect to all variables and setting to zero, we obtain the following:

$$\begin{aligned} & \mathbf{w} = \sum_{i=1}^{N}\alpha_i\boldsymbol{\varphi}_i \\ & \sum_{i=1}^{N}\alpha_i = 0 \\ & e_i = y_i - \left(\mathbf{w}^T\boldsymbol{\varphi}_i\right) + b, \quad \text{for } i = 1\cdots N \\ & \alpha_i = \eta e_i, \quad \text{for } i = 1\cdots N \end{aligned} \tag{3.24}$$

  - Solution to $\boldsymbol{\alpha}$ and b:

    The solution to Eq. (3.24) can be readily derived as:

$$\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{K} + \frac{1}{\eta}\mathbf{I} & \mathbf{e}_N \\ \mathbf{e}_N^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \tag{3.25}$$

    where $\mathbf{I}$ is the identity matrix, $\mathbf{y} = [y_1 \cdots y_N]^T$ and $\mathbf{e}_N$ is an all-one vector with size N.

– Low rank approximation:
By using the Woodbury identity, we have:

$$\begin{bmatrix} \mathbf{K}+\frac{1}{\eta}\mathbf{I} & \mathbf{e}_N \\ \mathbf{e}_N^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{T}^{-1}\left(\mathbf{I}-\frac{N\mathbf{T}^{-1}}{\mathbf{e}_N^T\mathbf{T}^{-1}\mathbf{e}_N}\right) & \frac{\mathbf{T}^{-1}\mathbf{e}_N}{\mathbf{e}_N^T\mathbf{T}^{-1}\mathbf{e}_N} \\ \frac{\mathbf{e}_N^T\mathbf{T}^{-1}}{\mathbf{e}_N^T\mathbf{T}^{-1}\mathbf{e}_N} & -\frac{1}{\mathbf{e}_N^T\mathbf{T}^{-1}\mathbf{e}_N} \end{bmatrix} \tag{3.26}$$

where $\mathbf{T}=\mathbf{K}+\frac{1}{\eta}\mathbf{I}$. Therefore, Eq. (3.25) implies a complexity of $\mathcal{O}(N^3)$ for the matrix inversion, since the computation is dominated by $\mathbf{T}^{-1}$. Now we demonstrate how the complexity can be reduced by kernel approximation

$$\mathbf{K} \approx \mathbf{\Phi}_{\mathcal{D}}^T\mathbf{U}\mathbf{U}^T\mathbf{\Phi}_{\mathcal{D}} = k\left(\mathbf{X}_{\mathcal{D}},\mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}},\mathbf{X}_{\mathcal{D}}\right)$$

By once again applying the Woodbury,

$$\begin{aligned} & \left(\mathbf{K}+\frac{1}{\eta}\mathbf{I}\right)^{-1} \\ \approx\ & \left(k\left(\mathbf{X}_{\mathcal{D}},\mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}},\mathbf{X}_{\mathcal{D}}\right)+\frac{1}{\eta}\mathbf{I}\right)^{-1} \\ =\ & \eta\left(\mathbf{I}-\eta k\left(\mathbf{X}_{\mathcal{D}},\mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\left(\mathbf{I}+\eta\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}},\mathbf{X}_{\mathcal{D}}\right)k\left(\mathbf{X}_{\mathcal{D}},\mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\right)^{-1}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}},\mathbf{X}_{\mathcal{D}}\right)\right) \\ & \left\{\text{By applying the explicit feature map } \mathbf{B}\triangleq\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}},\mathbf{X}_{\mathcal{D}}\right)\in\mathbb{R}^{r\times N}\right\} \\ =\ & \eta\left(\mathbf{I}-\mathbf{B}^T\left(\frac{1}{\eta}\mathbf{I}+\mathbf{B}\mathbf{B}^T\right)^{-1}\mathbf{B}\right) \end{aligned} \tag{3.27}$$

where notations are referred to Eq. (3.8).
Eq. (3.27) shows that for given $\mathbf{U}\in\mathbb{R}^{q\times r}$, where $r\ll N\ll q$, the complexity of LS-SVM is reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}\left(\max(N,r^2)r\right)$.

- **Training Algorithm Option 2):** training in the primal:
  - with **Representer Theorem** (classification model f_a, no kernel approximation):
    - o The Representer Theorem: $\mathbf{w}=\sum_{i=1}^N\alpha_i\boldsymbol{\varphi}_i=\mathbf{\Phi}_{\mathcal{D}}\boldsymbol{\alpha}$ reformulates the objective function in Eq. (3.21) as:

$$\begin{aligned} l(\boldsymbol{\alpha},b) &= \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{\Phi}_{\mathcal{D}}^T\mathbf{\Phi}_{\mathcal{D}}\boldsymbol{\alpha}+\frac{\eta}{2}\sum_{i=1}^N\left(1-\left(\boldsymbol{\alpha}^T\mathbf{\Phi}_{\mathcal{D}}^T\boldsymbol{\varphi}_i+b\right)y_i\right)^2 \\ &= \frac{1}{2}\boldsymbol{\alpha}^T k\left(\mathbf{X}_{\mathcal{D}},\mathbf{X}_{\mathcal{D}}\right)\boldsymbol{\alpha}+\frac{\eta}{2}\sum_{i=1}^N\left(1-\left(\boldsymbol{\alpha}^T k\left(\mathbf{X}_{\mathcal{D}},\mathbf{x}_i\right)+b\right)y_i\right)^2 \end{aligned} \tag{3.28}$$

    - o Solution: Setting derivatives of Eq. (3.28) to zero with respect to $\boldsymbol{\alpha}$ and b, we obtain:

$$\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} = \begin{bmatrix}\mathbf{K}(\mathbf{K}+\frac{1}{\eta}\mathbf{I}) & \mathbf{K}\mathbf{e}_N \\ \mathbf{e}_N^T\mathbf{K} & N\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{K}\mathbf{y}\\ \mathbf{e}_N^T\mathbf{y}\end{bmatrix} \tag{3.29}$$

    where $\mathbf{K}=k\left(\mathbf{X}_{\mathcal{D}},\mathbf{X}_{\mathcal{D}}\right)\in\mathbb{R}^{N\times N}$.

– with **low rank approximation** (classification model f_b):

o For a given $\mathbf{U} = \mathbf{\Phi}_{\mathcal{G}}\mathbf{A}$, the low rank approximation leads to the expression

$$\mathbf{K} \approx \mathbf{\Phi}_{\mathcal{D}}^T\mathbf{U}\mathbf{U}^T\mathbf{\Phi}_{\mathcal{D}} = k\left(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{X}_{\mathcal{D}}\right). \tag{3.30}$$

The representation of the classifier is written as

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{U}\boldsymbol{\beta}_i = \mathbf{U}\mathbf{U}^T\mathbf{\Phi}_{\mathcal{D}}\boldsymbol{\alpha} = \mathbf{\Phi}_{\mathcal{G}}\mathbf{A}\mathbf{A}^T\mathbf{\Phi}_{\mathcal{G}}^T\mathbf{\Phi}_{\mathcal{D}}\boldsymbol{\alpha} = \mathbf{\Phi}_{\mathcal{G}}\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{X}_{\mathcal{D}}\right)\boldsymbol{\alpha}$$

The objective function in Eq. (3.21) becomes:

$$\begin{aligned} l(\boldsymbol{\alpha}, b) &= \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{\Phi}_{\mathcal{D}}^T\mathbf{\Phi}_{\mathcal{G}}\mathbf{A}\mathbf{A}^T\mathbf{\Phi}_{\mathcal{G}}^T\mathbf{\Phi}_{\mathcal{D}}\boldsymbol{\alpha} \\ &\quad + \frac{\eta}{2}\sum_{i=1}^{N}\left(1 - \left(\boldsymbol{\alpha}^T\mathbf{\Phi}_{\mathcal{D}}^T\mathbf{\Phi}_{\mathcal{G}}\mathbf{A}\mathbf{A}^T\mathbf{\Phi}_{\mathcal{G}}^T\boldsymbol{\varphi}_i + b\right)y_i\right)^2 \\ &= \frac{1}{2}\boldsymbol{\alpha}^T k\left(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{X}_{\mathcal{D}}\right)\boldsymbol{\alpha} \\ &\quad + \frac{\eta}{2}\sum_{i=1}^{N}\left(1 - \left(\boldsymbol{\alpha}^T k\left(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{x}_i\right) + b\right)y_i\right)^2 \end{aligned} \tag{3.31}$$

o Solution: Due to the approximation in Eq. (3.30), the solution presented in Eq. (3.29) becomes

$$\begin{bmatrix}\boldsymbol{\alpha} \\ b\end{bmatrix} = \begin{bmatrix}\mathbf{K}_{\beta}(\mathbf{K}_{\beta} + \frac{1}{\eta}\mathbf{I}) & \mathbf{K}_{\beta}\mathbf{e}_N \\ \mathbf{e}_N^T\mathbf{K}_{\beta}^T & N\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{K}_{\beta}\mathbf{y} \\ \mathbf{e}_N^T\mathbf{y}\end{bmatrix} \tag{3.32}$$

where $\mathbf{K}_{\beta} = k\left(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{X}_{\mathcal{D}}\right)$.
Similar to Eq. (3.26) and Eq. (3.27), by using the Woodbury identity, the dominating computation can be reduced thanks to this approximation, which leads to an overall complexity of order $\mathcal{O}\left(\max(N, r^2)r\right)$, for $N \gg r$.

– with **explicit feature map** (classification model f_c):

o Given the explicit feature vectors:

$$\boldsymbol{\beta}_i = \mathbf{A}^T\mathbf{\Phi}_{\mathcal{G}}^T\boldsymbol{\varphi}_i = \mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{x}_i\right),$$

we can construct an orthonormal matrix $\mathbf{V} = [\mathbf{v}_i, \cdots, \mathbf{v}_r]$, where $\mathbf{v}_i \in \mathbb{R}^r$, $i = 1 \cdots r$. The objective function in Eq. (3.21) becomes:

$$\begin{aligned} l(\mathbf{w}, b) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\eta}{2}\sum_{i=1}^{N}\left(1 - \left(\mathbf{w}^T\left(\mathbf{V}^T\mathbf{A}^T\mathbf{\Phi}_{\mathcal{G}}^T\boldsymbol{\varphi}_i\right) + b\right)y_i\right)^2 \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\eta}{2}\sum_{i=1}^{N}\left(1 - \left(\mathbf{w}^T\left(\mathbf{V}^T\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{x}_i\right)\right) + b\right)y_i\right)^2 \end{aligned} \tag{3.33}$$

where $\mathbf{w} \in \mathbb{R}^r$. Hence, we are solving a r dimensional linear classification problem.

Remark. Note that we can also use

$$l(\tilde{\mathbf{w}}, b) = \frac{1}{2}\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} + \frac{\eta}{2}\sum_{i=1}^{N}\left(1 - \left(\tilde{\mathbf{w}}^T\left(k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{X}_{\mathcal{G}}\right)\mathbf{A}\mathbf{A}^T k\left(\mathbf{X}_{\mathcal{G}}, \mathbf{x}_i\right)\right) + b\right) y_i\right)^2 \tag{3.34}$$

with $\tilde{\mathbf{w}} \in \mathbb{R}^{|\mathcal{G}|}$, $|\mathcal{G}| \geq r$, to avoid the computation for constructing the basis matrix $\mathbf{V}$.

- Solution: Let $\mathbf{B} = \begin{bmatrix}\boldsymbol{\beta}_1 & \cdots & \boldsymbol{\beta}_N\end{bmatrix}$,

$$\begin{bmatrix}\mathbf{w} \\ b\end{bmatrix} = \begin{bmatrix}\mathbf{B}\mathbf{B}^T + \frac{1}{\eta}\mathbf{I} & \mathbf{B}\mathbf{e}_N \\ \mathbf{e}_N^T\mathbf{B}^T & N\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{B}\mathbf{y} \\ \mathbf{e}_N^T\mathbf{y}\end{bmatrix} \tag{3.35}$$

  where we can directly see that the computational complexity is of order $\mathcal{O}\left(\max(N, r^2)r\right)$.

This example shows the relation between the three different *classification models* f_a, f_b and f_c, where f_a directly follows the Representer Theorem; f_b results from the low rank approximation using a subspace model for a given basis matrix $\mathbf{U}$; and f_c is the linear classifier using explicit feature vectors. The *classification learning objective* is the LS-SVM and the three classification models lead to different formulations and computational complexities. The dual and primal *training algorithms* are both presented. Note that from the viewpoint of simplicity, it is more natural to train the classifier with f_b in the dual, but with f_c in the primal. We can see that f_b results in a lower complexity compared to f_a in both primal and dual cases.

3.5 Conclusion

Given training data, *kernel approximation* techniques are typically needed for reducing the computational complexity. This approximation can be interpreted using a *subspace model in the RKHS* (c.f. Eq. (3.8) and Eq. (3.17)), which further implies the possibility of using an *explicit feature vector* (c.f. Eq. (3.12)) instead of the kernel matrix. When using explicit feature vectors, we are moving away from the RKHS, but instead working in a high dimensional Euclidean space. Linear models and algorithms are then applied at the classification training step. The advantage is that one does not need to reformulate/approximate linear techniques using their kernelized counterpart, which leads to a simpler modeling procedure. However, some might find it a bit unnatural, since RKHS is indeed what provides us all the elegant and handy properties. Nevertheless, one should keep in mind that the explicit feature vector is a result of the very existence of the RKHS.

Furthermore, different representations in Eq. (3.17) result in distinct *classification models* and *learning objectives*, which then lead to various modeling simplicity, robustness and computational complexity. This is illustrated in both Sec. 3.4 as a general principal (c.f. Table 3.2 and Fig. 3.2) and in Sec. 3.4.3 using LS-SVM as a concrete example.

Chapter 4

Summary of Included Papers

In this thesis, we have developed kernel methods for classification tasks using subspace models in the Reproducing Kernel Hilbert Space (RKHS). The included papers are summarized in this chapter. The contribution is divided into two sections with respect to different subspace models:

- Kernel subspace model (Sec. 4.1): Given a kernel function, we assume that data vectors span a subspace in the RKHS.

- Class-specific subspace model (Sec. 2.2): Data from different classes are assumed to span an individual subspace in a Hilbert space that is induced by a class-specific kernel function.

An overview of this chapter and its relation to Chapter 3 can be found in Fig. 4.1. According to this categorization, a list of included papers and their summaries are shown as follows:

- Kernel subspace model:
  - **Paper 1. Kernel Successive Orthogonal Subspace Analysis:** A RKHS subspace transformation technique for maximizing the ratio between the between-class data separation and within-class data separation.
- CLAss-Specific Subspace Kernel (CLASK) model:
  - **Paper 2. Learning Hierarchical Feature Space Using CLAss-specific Subspace Multiple Kernels - Metric Learning for Classification:** A model selection technique that chooses kernel types and parameters from a given set of kernel functions using class-specific multiple kernels.
  - **Paper 3. Enhanced Distance Subset Approximation Using CLAss-specific Subspace Kernel for Supervised Learning:** A sampling technique for kernel approximation using class-specific subspace data model, which is aiming at enhancing the between-class distance.

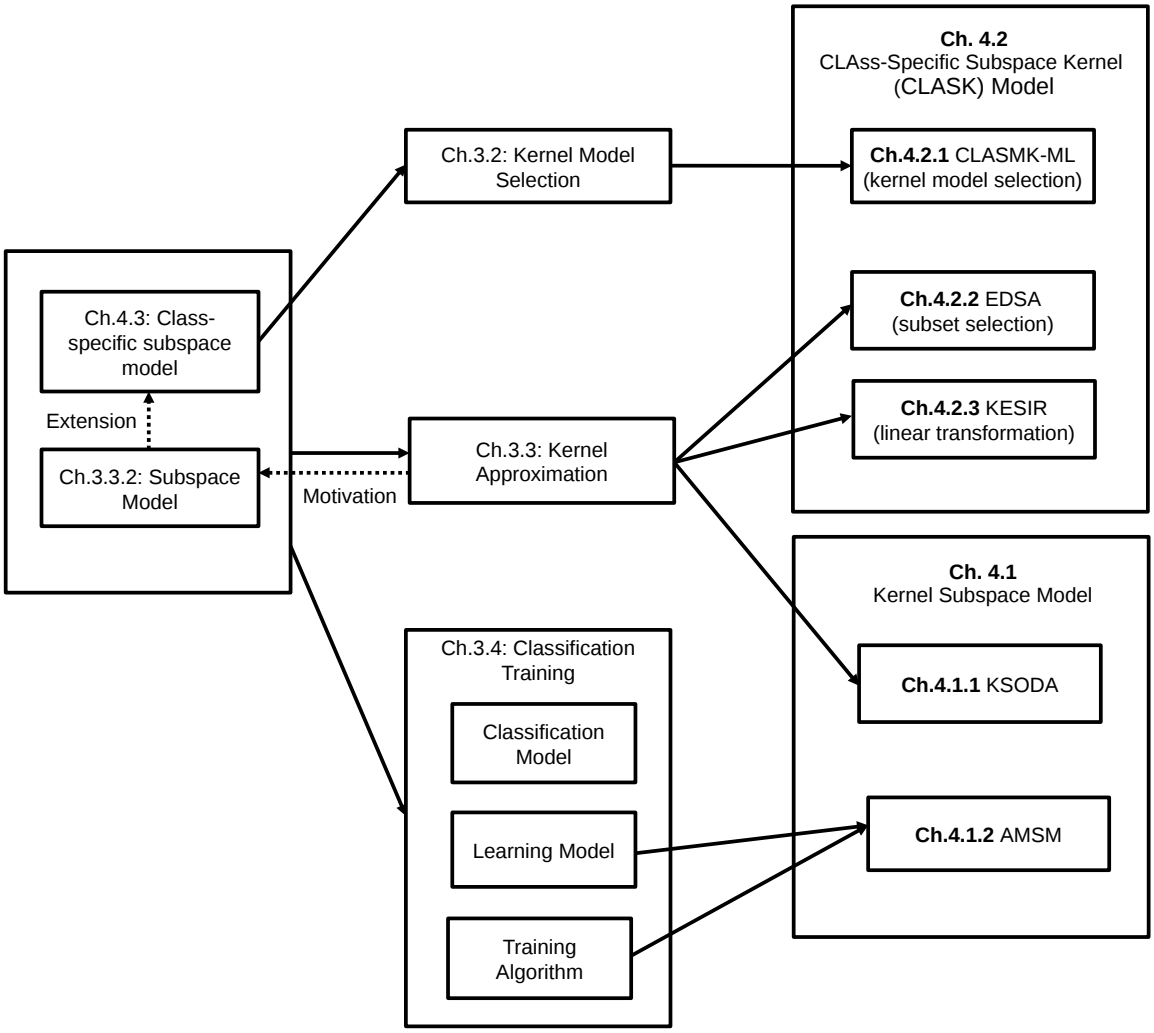

Figure 4.1: The organization of this chapter and the reference to Chapter 3.

– **Paper 4. Kernel Subspace Empirical Intersection Removal for Kernel Approximation and Classification:** A metric learning technique is proposed for kernel approximation using CLASK model. The goal is to find class-specific subspaces that have large canonical distance.

– **Paper 5. CLAss-specific Subspace Kernel Representations and Adaptive Margin Slack Minimization for Large Scale Classification:** In this paper, a classification framework for large scale training is proposed based on 1) the CLASK model, and 2) a novel data selection scheme using an adaptive margin with sequential and parallel framework.

4.1 Kernel Subspace Model

4.1.1 Overview

The kernel subspace model (c.f. Eq. (3.7)) is essentially the underlying assumption behind kernel approximation, where we assume that the data generating mechanism follows a subspace structure in the RKHS $\mathcal{F}$ that is common for all classes. This subspace is then used for low rank approximation and the quality of the approximation is evaluated by the reconstruction error. Based on this model assumption, we developed the following techniques for classification tasks.

4.1.2 Included Papers

Paper 1: Kernel Successively Orthogonal Discriminative Analysis (SODA)

The Linear Discriminative Analysis (LDA) and its kernel counterpart KLDA are popular for solving classification problems, due to its intuitive formulation, theoretical properties and the closed form solution. The goal is to project the data points onto a low dimensional subspace, where the ratio between the between-class scattering and within-class scattering is maximized. However, one drawback is that given C classes, LDA can only find a $C-1$ dimensional subspace. In this work, we have tackled this discriminative learning problem using an recursive formulation called Kernel Successively Orthogonal Analysis (KSODA) and found the optimal solution by an iterative algorithm. KSODA can be applied as a RKHS transformation technique for kernel approximation in classification tasks.

4.2 CLAss-specific Subspace Kernel (CLASK) Function for Classification

4.2.1 Class-specific Subspace Model

The classical kernel approximation techniques are based on the kernel subspace model presented in the previous section. However, in many applications, one subspace is not adequate to describe the intrinsic data structure. Hence, some machine learning techniques, such as subspace clustering [51, 52, 53] and subspace classifiers [54, 55, 56, 57] are based on a multiple-subspace model.

In particular, in a classification problem, one is interested in finding subspaces that capture complex intrinsic data structures, while having discriminative ability for different classes.

To this end, we adopt a class-specific subspace model. More precisely, given data vector $\boldsymbol{\varphi} \in$ class $c \in \{1 \cdots C\}$, the model assumption is as follows:

$$\boldsymbol{\varphi} = \mathbf{U}_c \boldsymbol{\beta} + \mathbf{e} \tag{4.1}$$

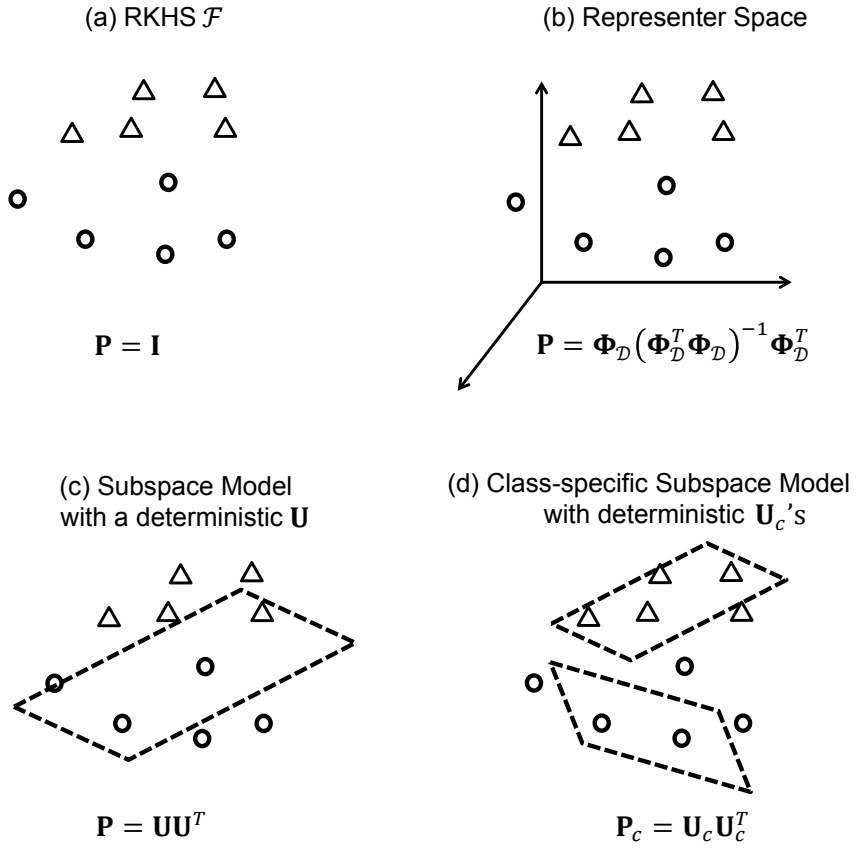

Figure 4.2: Four projections are illustrated.

where matrix $\mathbf{U}_c \in \mathbb{R}^{p\times m_c}$ has orthonormal column vectors, $\boldsymbol{\beta} \in \mathbb{R}^{m_c}$ and $\mathbf{e} \perp \mathbf{U}_c$ is the vector containing residuals. Moreover, in addition to the subspace structure for different classes, we assume that individual kernel functions may apply.

An illustration can be found in Fig. 4.2. In Fig. 4.2 (b), given a set of training data $\mathcal{D}$ and the Structural Risk Minimization (SRM) [9] framework, the Representer theorem [9] states that the optimal function f^* learned from $\mathcal{D}$ can be written as a linear combination of all training data, i.e. $f^* \in col(\mathbf{\Phi}_{\mathcal{D}})$. It means that without low rank approximation, the optimization problem for finding an optimal classifier given $\mathcal{D}$ is equivalent to projecting all training data onto $col(\mathbf{\Phi}_{\mathcal{D}})$. Fig. 4.2 (c) interprets the classical low rank approximation using the subspace model in Eq. (3.7), whereas Fig. 4.2 (d) illustrates how the subspace model in Eq. (4) results in a different setup compared to Fig. 4.2 (c).

4.2.2 CLAss-Specific Kernel (CLASK) Function

Given the class-specific subspace model in Eq. (4), we define the CLAss-Specific Kernel (CLASK) function as follows.

Definition 4.1 (CLAss-Specific Kernel (CLASK) function:)**.** Given C classes and an ordered set of kernel functions $\{k_1, \cdots, k_C\}$, the CLASK function is a function

$h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined as:

$$h(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{c=1}^{C} k_c(\mathbf{x}, \tilde{\mathbf{x}}) \tag{4.2}$$

for any $\mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X}$

4.2.3 Feature Map

Given the class-specific subspace model in Eq. (4) and the CLASK function in Definition 3.1, let $\mathcal{F}^C$ denote the Hilbert Space induced by CLASK. For a data vector $\mathbf{x} \in$ class c, one feature map $\varphi : \mathcal{X} \to \mathcal{F}^C$ can be defined as:

$$\begin{aligned} \varphi(\mathbf{x}) &= \begin{bmatrix} k_1(\mathbf{x}, \cdot) \\ \vdots \\ k_C(\mathbf{x}, \cdot) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{U}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{U}_C \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_C \end{bmatrix} + \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_C \end{bmatrix} \\ &= \mathbf{U}\boldsymbol{\beta} + \mathbf{e} \end{aligned} \tag{4.3}$$

where $\{k_1, \cdots, k_C\}$ are the given kernel functions from Definition 3.1; $\boldsymbol{\beta}_r$ and $\mathbf{e}_r$ are the weight vector and the residual associated with subspace basis $\mathbf{U}_r \in \mathbb{R}^{p \times m_r}$.

Given the model assumption Eq. (4), the subsequent questions are:

- How to incorporate the class-specific subspace model into the framework of kernel methods?
- When estimating unknown subspaces, what desired properties are we looking for?

4.2.4 Included Papers

Motivated by the class-specific subspace model, we have developed a sequence of learning techniques that attempt to answer these questions. A brief summary is presented in the following papers.

Paper 2: CLASMK-ML for Kernel Model Selection

Kernel model selection is one of the major issues for kernel techniques, since the kernel function completely determines the similarity (i.e. the inner product) defined on the RKHS. In this paper, we have proposed a series of optimization criteria based on the class-specific subspace model to automatically select the optimal kernel function from a given finite countable set. There are three scenarios: 1) one unified optimal kernel function is identified for all classes; 2) different optimal kernel functions are allowed for each individual class; and 3) Multiple-Kernels (MK) with different weights

are selected for different classes. Cases 2) and 3) are proposed based on a kernel function that enables the possibility of describing data from different categories using their own kernel functions. The learning process is based on a metric learning framework and the corresponding techniques are called the CLAss-specific Subspace Kernel- Metric Learning (CLASK-ML) and CLAS-Multiple-Kernel- Metric Learning (CLASMK-ML).

A hierarchical learning structure is also proposed to improve the classification performance for a given base classifier by feature augmentation. As a future direction, tests using more types of kernel functions are under progress. Moreover, feature pruning strategies are needed at each layer for large scale datasets. We are also investigating the possibilities of integrating the feature augmentation technique into a deep kernel network.

Paper 3: EDSA Sampling Technique for Kernel Approximation

In this paper, by assuming the class-specific subspace model, we propose a novel kernel function called the CLAss-specific Subspace Kernel, which is incorporated into the kernel approximation framework. Given an ordered set of kernel functions $\{k_1, \cdots, k_C\}$ and the underlying model assumption Eq. (4), CLASK is associated to a feature space $\mathcal{F}^C$, such that $\left[k_1(\mathbf{x}, \cdot)^T, \cdots, k_C(\mathbf{x}, \cdot)^T\right]^T \in \mathcal{F}^C$ for all $\mathbf{x} \in \mathcal{X}$. The idea is to estimate a subspace spanned by $k_c(\mathbf{x}, \cdot)$, $\forall \mathbf{x} \in$ class c, for every $c \in \{1, \cdots, C\}$. The final basis matrix is then described using $\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & \mathbf{U}_C \end{bmatrix}$, where $\mathbf{U}_c$ is the estimated subspace basis matrix for class c. Moreover, when restricted to the subspace $col(\mathbf{U})$, a probability bound on the class separability is proposed as a function of both the between-class projection and the within-class projection. To achieve this goal, an optimization criterion is formulated to find the most appropriate subspace matrices. Due to the high computational complexity, the solution is approximated by a sampling technique called the Enhanced Distance Subset Approximation (EDSA).

Paper 4: Kernel Empirical Subspace Intersection Removal (ESIR)

In standard subspace classification techniques, an individual subspace basis is learned for each class in the training data. For an unlabeled testing pattern, the classification criterion is to identify the subspace corresponding to the smallest projection distance. This criterion provides a natural measure of the discriminative power for each individual subspace. Particularly, from a statistical learning point of view, for a given classification accuracy on the training data, the complexity of the subspaces (which is characterized by the VC dimension) influences the generalization ability of the subspace classifier. That is, with a high model complexity, the classifier is prone to

overfitting. To obtain a high generalization performance, one strategy is to reduce the subspace dimension of the learning model without sacrificing the training accuracy. In this paper, we are aiming at learning such basis matrices for kernel approximation based on the aforementioned criteria. The goal is to select as few subspace directions as possible, which provide the best generalization ability on the class discrimination. This can be achieved using metric learning techniques, which usually require the performance evaluation on the training set. On the contrary, our method only demands the computations on the subspace bases, which is a much smaller set compared to the training data. After learning the subspace bases, they can be used for kernel approximation and classification with any base classifiers.

Paper 5: CLASK Feature Extraction and Large Scale Classification

The contribution of this paper is the development of classification frameworks for large-scale problems, which is based on a novel classification training technique called the Adaptive Margin Slack Minimization (AMSM) that employees an adaptive data selection scheme using LS-SVM as its base classifier.

LS-SVM is a simple but efficient classification technique, where the separating hyperplane is identified by performing linear regression on the training data using their closest marginal hyperplane [58]. L2-SVM, when training on the primal given the Representer theorem (c.f. Theorem 2.7), can be considered as an extension to LS-SVM, where only the marginally misclassified patterns are included for training at each iteration. In this paper, we further extend the idea of subset selection using an adaptive margin. The margin is selected based on a greedy mechanism such that convergence is guaranteed.

Furthermore, we have explored the possibilities of using the kernel function CLASK as the feature extractor in combination with the proposed AMSM algorithm.

To handle large-scale datasets, we have proposed two frameworks: the Memory Efficient Sequential Processing (MESP) framework and the Parallel Sequential Processing (PSP) framework, which can be efficiently implemented using a queue and a stack data structure, respectively. The framework MESP can be employed when only one processor is available, whereas PSP is efficient for multithread or distributed processing.

4.2.5 A Feature Transformation System Using CLASK

In Fig. 4.3, we have shown a feature transformation system using the kernel function CLASK. It also serves as a summary of the techniques proposed in the aforementioned papers. The output of the system is a finite dimensional vector space that can be used as the new feature space.

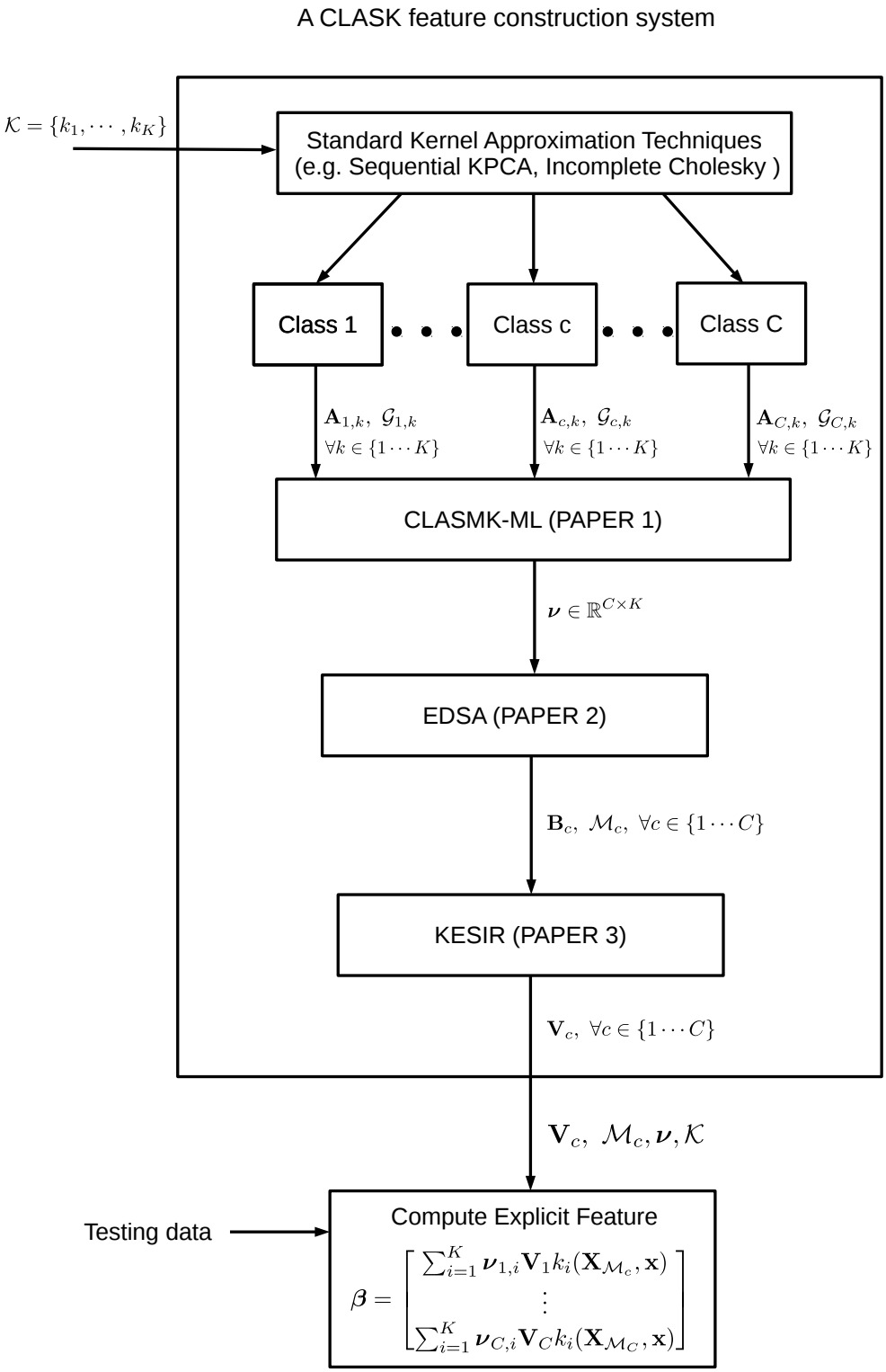

Figure 4.3: This figure shows how the techniques presented in the included papers are aggregated into a feature transformation system using the kernel function CLASK.

4.3 Software Package: DeepCLASK

A MATLAB package called DeepCLASK and its documentation can be found in the following link: `https://github.com/yinan16/DeepCLASK`.

The package is an implementation (with some extensions) of the aforementioned algorithms developed in this thesis work. A brief summary of the software structure is illustrated in Fig. 4.4.One may refer to the website for further demonstrations, documentations and future updates.

4.4 Conclusions

Kernel methods became popular due to the success of Support Vector Machines in various domains, where a nonlinear map is implicitly constructed by the definition of a positive semi-definite kernel function. Although it is mostly known that kernels are closely tied to SVMs, one shall not neglect the fact that kernel techniques can be used as feature transformation for gaining better data descriptive power.

However, kernel methods usually involve pairwise evaluation on the training set. When the training size N grows large, the operations and storage needed for the $N \times N$ kernel matrix might become prohibitive.

To overcome such issues, kernel approximation techniques can be applied, which gives rise of the subspace model in the Kernel Reproducing Hilbert Space. Furthermore, for classification problems, one aims at enhancing the class-separability with approximate solutions, where the class-separability can be defined in various ways with essentially two elements: the within-class distance and the between-class distance. With optimization techniques such as metric learning, one can approximate the optimality by enhancing the within-class distance, while suppressing the between-class distance.

In this thesis, we first explored the underlying structure of kernel subspace models and proposed the CLAss-specific Subspace Kernel function for kernel approximation and classification. Given the CLASK function, a kernel feature transformation system is presented. The system consists of three parts: 1) Multiple-Kernel model selection; 2) Subset selection for kernel approximation; and 3) Linear transformation in the RKHS, where the class-separability is enhanced at each step. Moreover, the algorithms are designed to handle large scale problems on two levels. First of all, from a algorithmic point of view, given C classes, the order of the computational complexity of the class-specific subset selection and learning individual transformation matrices are both scaled down by a factor of C^2 compared to the one kernel subspace model. Secondly, large scale computational frameworks are proposed for sequential learning and parallelization, which can be then used with distributed computing infrastructures.

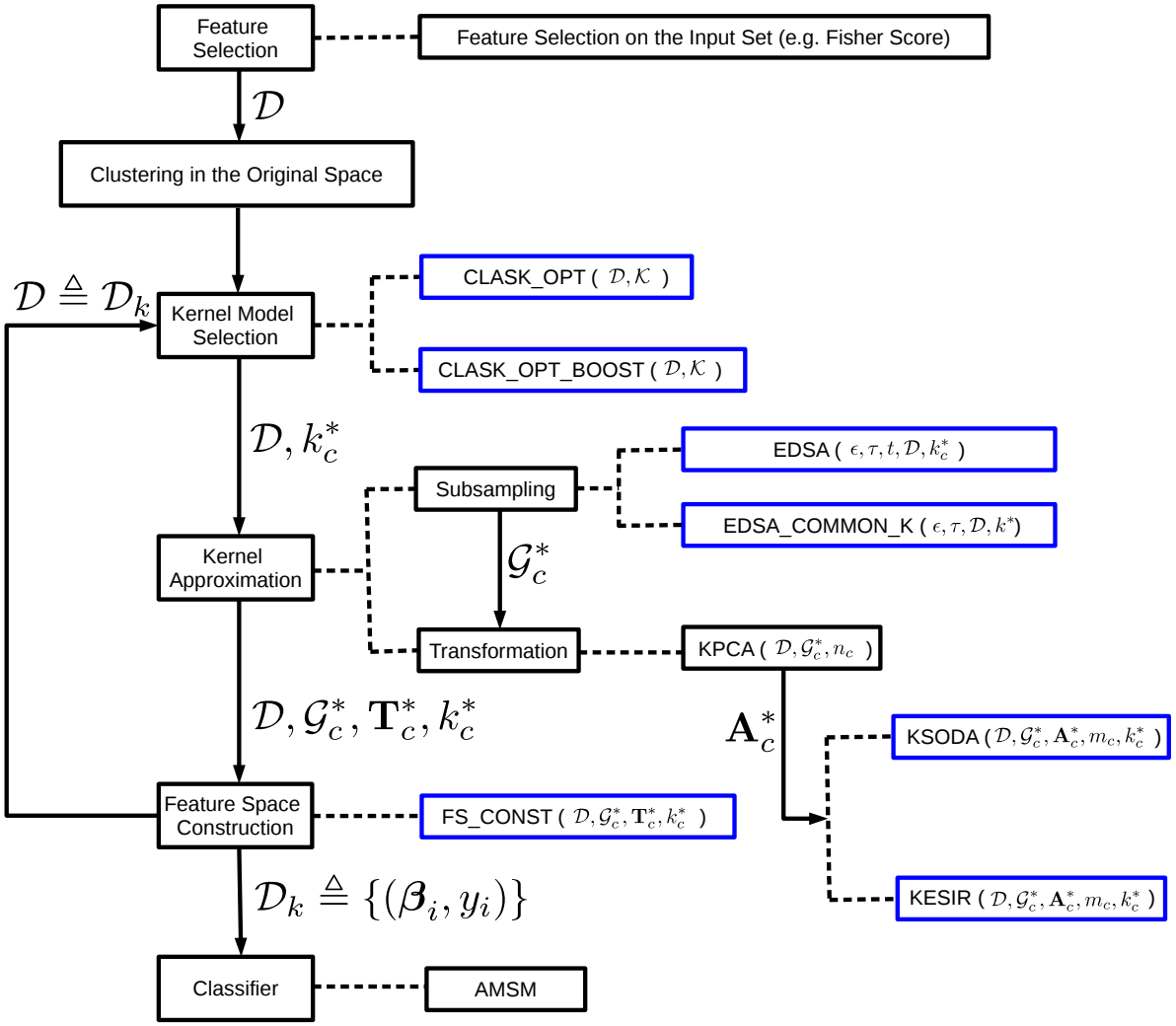

Figure 4.4: The training framework of the package DeepCLASK. This figure shows the main functions (indicated by blue windows) contained in DeepCLASK. The loop indicates the multilayer structure of the learning model.

Chapter 5

Future work

In this thesis, we have explored the class-specific subspace model in the RKHS using class-specific kernel functions for feature transformation and kernel approximation. Besides promising results within the scope of this thesis, there are three main directions yet to be investigated. Firstly, one future direction would be to apply our feature extraction system (c.f. Fig. 4.3) with the CLASK model to various applications using domain specific kernel functions, such as natural language processing, image/video processing, medical applications, etc. Secondly, with the growing needs for big data processing capabilities and the development of distributed and parallel computing frameworks, it is important to adapt our algorithms to the state-of-the-art software architectures for solving large scale learning problems. The scalability can be achieved in different ways. One way to look at the problem is to analyze the stochastic properties of the learning system with respect to random training sets. For instance, when using a "divide-and-conquer" type of learning framework, the consistency of the learning model and the training algorithm plays a key role in the learnablility of the system. On the other hand, some approaches (such as utilization of Graphic Processing Units) use a large amount of small computing units for speeding up the process. In this case, adaptations of our algorithms can be made in order to benefit from such frameworks. Lastly, further testing of the system as a whole is to be conducted in order to establish its strengths and boundaries.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning.* Cambridge, U.K.: Springer, 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data mining, Inference, and Prediction.* New York: Springer, 2009.

[3] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35(8), pp. 1798–1828, 2013.

[4] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22(1), pp. 4–37, 2002.

[5] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15(2), pp. 262–286, 2006.

[6] A. d'Aspremont, L. Ghaoui, M. Jordan, and G. R. G. Lanckriet, "A direct formulation for sparse PCA using semidefinite programming," *SIAM Review*, vol. 49(3), pp. 434–448, 2007.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1967, pp. 281–297.

[8] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering," *Machine Learning*, vol. 75, pp. 245–249, 2009.

[9] B. Schölkopf and A. Smola, *Learning with Kernels.* MIT Press, 2002.

[10] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* New York, NY, USA: Cambridge University Press, 2004.

[11] S. Y. Kung, *Kernel Methods and Machine Learning.* Cambridge Press, 2014.

[12] S. Theodoridis, *Machine Learning, A Bayesian and Optimization Perspective.* Academic Press, 2015.

[13] V. N. Vapnik, *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[14] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 4th ed. Academic Press, 2009.

[15] C. Cortes and V. N. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20(3), 1995.

[16] L. Vandenberghe and S. P. Boyd, *Convex Optimization*. Stanford University, 2004.

[17] [Online]. Available: https://en.wikipedia.org/wiki/Occam%27s_razor

[18] G. Folland, *Real Analysis: Modern Techniques and Their Applications*, 2nd ed. Wiley, 1999.

[19] D. Montgomery, "Non-separable metric spaces," *Fundamenta Mathematicae*, vol. 1(25), pp. 527–533, 1935.

[20] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, vol. 68(3), pp. 337–404, 1950.

[21] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, pp. 419–444, 2002.

[22] C. Corinna, M. Mehryar, and R. Afshin, "Algorithms for learning kernels based on centered alignment," *Journal of Machine Learning Research*, vol. 13, pp. 795–828, 2012.

[23] M. G. Genton, N. Cristianini, J. Shawe-taylor, and R. Williamson, "Classes of kernels for machine learning: a statistics perspective," *Journal of Machine Learning Research*, vol. 2, pp. 299–312, 2001.

[24] K. Ghiasi-Shirazi, R. Safabakhsh, and M. Shamsi, "Learning translation invariant kernels for classification," *Journal of Machine Learning Research*, pp. 1353–1390, 2010.

[25] G. Arfken, *Mathematical Methods for Physicists*, 3rd ed. Orlando, FL: Academic Press, 1985.

[26] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*, 1st ed. Springer NY, 2004.

[27] G. Carlos, "Lecture notes," *Machine Learning Course*, 2007. [Online]. Available: http://www.cs.cmu.edu/~guestrin/Class/15781/slides/learningtheory-bns-annotated.pdf

[28] R. Picard and D. Cook, "Cross-validation of regression models," *Journal of the American Statistical Association*, vol. 79(387), pp. 575–583, 1984.

[29] S. Geisser, *Predictive Inference*. NY: Chapman and Hall, 1993.

[30] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," *BMC Bioinformatics*, vol. 7(91), 2006.

[31] M. Gönen and E. Alpaydin, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[32] A. Hoerl and R. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, pp. 80–86, 1970.

[33] W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31(2), pp. 221–239, 1989.

[34] K. Diamantaras and S. Y. Kung, *Principal Component Neural Networks*. John Wiley, 1996.

[35] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10(5), pp. 1299–1319, 1998.

[36] ——, "Kernel principal component analysis," *Advances in Kernel Methods - Support Vector Learning*, pp. 327–352, 1999.

[37] W. Rudin, *Fourier Analysis on Groups*. Wiley-Interscience New York, 1994.

[38] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, 2007.

[39] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, vol. 13, 2001, pp. 682–688.

[40] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26(2), pp. 214–225, 2004.

[41] P. Drineas and M. Mahoney, "On the Nyström method for approximating a gram matrix for improved kernelbased learning," *Journal of Machine Learning Research*, pp. 2153–2175, 2005.

[42] K. Zhang, I. Tsang, and J. Kwok, "Improved Nyström low-rank approximation and error analysis," in *International Conference on Machine Learning*, 2008, pp. 1232–1239.

[43] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the Nyström method," *Journal of Machine Learning Research*, vol. 13, pp. 981–1006, 2012.

[44] T. Yang, Y. Li, M. Mahdavi, R. Jin, and Z. Zhou, "Nyström method vs random Fourier features: a theoretical and empirical comparison," in *Advances in Neural Information Processing Systems*, 2012.

[45] M. Ouimet and Y. Bengio, "Greedy spectral embedding," in *Proceeding of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005, pp. 253–260.

[46] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *Journal of Machine Learning Research*, vol. 2, pp. 243–264, 2001.

[47] F. Bach and M. Jordan, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2002.

[48] ——, "Predictive low-rank decomposition for kernel methods," in *Proceeding of International Conference on Machine Learning*, 2005.

[49] S. Si, C. J. Hsieh, and I. Dhillon, "Memory efficient kernel approximation," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 701–709.

[50] J. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods*, pp. 185–208, 1999.

[51] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31(2), pp. 210–227, 2009.

[52] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[53] V. Zografos, L. Ellis, and R. Mester, "Discriminative subspace clustering," in *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, 2013, pp. 2107–2114.

[54] E. Oja, *Subspace Methods of Pattern Recognition*. Research Studies Press, Letchworth and J. Wiley, 1983.

[55] L. Scharf and B. Friedlander, "Matched subspace detectors," *Signal Processing, IEEE Transactions on*, pp. 2146–2157, 1994.

[56] S. Watanabe, P. F. Lambert, C. Kulikowski, J. Buxton, and R. Walker, "Evaluation and selection of variables in pattern recognition," *Computer and Information Sciences*, vol. 2, pp. 91–122, 1967.

[57] K. Tsuda, "Subspace classifier in the Hilbert space," *Pattern Recognition Letters*, vol. 20, pp. 513–519, 1999.

[58] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9(3), pp. 293–300, 1999.