NEW OPTIMIZATION METHODS AND APPLICATIONS IN KERNEL-BASED
MACHINE LEARNING

By

ONUR ŞEREF

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2006
to the memory of my father,

Abdullah Şeref
ACKNOWLEDGEMENTS

First, I would like to thank my supervisory committee chair Dr. Panos Pardalos for his continuous encouragement and guidance. Dr. Pardalos provided many opportunities that allowed me to become a self-confident and independent academician. I would like to extend my gratitude to my other committee members Dr. Ravindra Ahuja, Dr. Edwin Romeijn and Dr. Tamer Kahveci for their valuable feedback. I would like to include all Industrial and Systems Engineering (ISE) faculty, especially Dr. Suleyman Tufekci and Dr. Cole Smith for their help. I am very thankful for the great contribution from Dr. Mingzhou Ding and other researchers from his lab in the Biomedical Engineering Department. I would like to thank all my colleagues in the Center for Applied Optimization Lab for their support, especially Erhun Kundakcioglu for his extra help.

My beloved wife Michelle has been my biggest support with her caring love and my daily inspiration, to whom I owe my eternal gratitude. I want to thank my mother Esma, my sister Ezgi and my brother Kıvanç whose love and encouragement brought me to Gainesville and helped me get my Ph.D. I also would like to thank Michelle’s family, Magdi, Roblyn and Timothy, for their support and love.

I reserve my most special appreciation for my beloved father Abdullah Şeref, who recently passed away. His honesty, humility, philosophy and his passion for science have always guided me and always will.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>7</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>8</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>10</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>12</td>
</tr>
<tr>
<td>1.1 Kernel-Based Machine Learning</td>
<td>14</td>
</tr>
<tr>
<td>1.2 Applications to Neuroscience</td>
<td>15</td>
</tr>
<tr>
<td>1.3 Regularized Generalized Eigenvalue Classifiers</td>
<td>17</td>
</tr>
<tr>
<td>2 KERNEL-BASED LEARNING METHODS</td>
<td>20</td>
</tr>
<tr>
<td>2.1 Kernels</td>
<td>20</td>
</tr>
<tr>
<td>2.2 Generalization Theory</td>
<td>24</td>
</tr>
<tr>
<td>2.3 Optimization Theory</td>
<td>26</td>
</tr>
<tr>
<td>2.4 Support Vector Machines</td>
<td>30</td>
</tr>
<tr>
<td>2.4.1 Maximal Margin Classifier</td>
<td>31</td>
</tr>
<tr>
<td>2.4.2 Soft Margin Classifier</td>
<td>34</td>
</tr>
<tr>
<td>2.5 Gradient Based Adaptive Scaling</td>
<td>39</td>
</tr>
<tr>
<td>2.6 Support Vector Regression</td>
<td>40</td>
</tr>
<tr>
<td>2.6.1 Quadratic $\varepsilon$-Sensitive Loss</td>
<td>40</td>
</tr>
<tr>
<td>2.6.2 Linear $\varepsilon$-Sensitive Loss</td>
<td>42</td>
</tr>
<tr>
<td>3 SELECTIVE KERNEL-BASED METHODS</td>
<td>44</td>
</tr>
<tr>
<td>3.1 Selective Support Vector Machines</td>
<td>44</td>
</tr>
<tr>
<td>3.1.1 Selective 2-Norm Support Vector Machines</td>
<td>48</td>
</tr>
<tr>
<td>3.1.2 Selective 1-norm Support Vector Machines</td>
<td>51</td>
</tr>
<tr>
<td>3.1.3 Selecting Pattern Vectors with Large Margin</td>
<td>53</td>
</tr>
<tr>
<td>3.2 Selective Support Vector Regression</td>
<td>54</td>
</tr>
<tr>
<td>3.2.1 Selective 2-norm $\varepsilon$-Insensitive Regression</td>
<td>57</td>
</tr>
<tr>
<td>3.2.2 Selective 1-norm $\varepsilon$-Insensitive Regression</td>
<td>60</td>
</tr>
<tr>
<td>4 KERNEL METHODS APPLIED TO NEURAL DATA</td>
<td>64</td>
</tr>
<tr>
<td>4.1 Visual Discrimination Task</td>
<td>64</td>
</tr>
<tr>
<td>4.2 Methods</td>
<td>67</td>
</tr>
<tr>
<td>4.3 Classification and Channel Selection Results</td>
<td>67</td>
</tr>
<tr>
<td>4.4 Time Series Alignment with Dynamic Time Warping</td>
<td>72</td>
</tr>
<tr>
<td>4.5 Selective Support Vector Machines on Neural Data</td>
<td>74</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>Different stages of the visuomotor task</td>
<td>68</td>
</tr>
<tr>
<td>4-2</td>
<td>Channel names and locations for TI, LU and GE</td>
<td>69</td>
</tr>
<tr>
<td>5-1</td>
<td>Classification accuracy using linear kernel.</td>
<td>92</td>
</tr>
<tr>
<td>5-2</td>
<td>Classification accuracy using gaussian kernel.</td>
<td>93</td>
</tr>
<tr>
<td>5-3</td>
<td>Elapsed time in seconds using linear kernel.</td>
<td>93</td>
</tr>
<tr>
<td>5-4</td>
<td>Elapsed time in seconds using gaussian kernel.</td>
<td>93</td>
</tr>
<tr>
<td>5-5</td>
<td>Execution times for the parallel implementation</td>
<td>97</td>
</tr>
<tr>
<td>5-6</td>
<td>Efficiency for the parallel implementation</td>
<td>97</td>
</tr>
<tr>
<td>5-7</td>
<td>Datasets characteristics</td>
<td>105</td>
</tr>
<tr>
<td>5-8</td>
<td>Classification accuracy for ReGEC, I-ReGEC and SVM</td>
<td>106</td>
</tr>
<tr>
<td>5-9</td>
<td>Incremental dataset using I-ReGEC and percentage of the training set</td>
<td>106</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2-1</td>
<td>Kernel mapping of circular data in 2-D to 3-D.</td>
<td>21</td>
</tr>
<tr>
<td>2-2</td>
<td>Maximal margin classifier</td>
<td>32</td>
</tr>
<tr>
<td>2-3</td>
<td>Soft margin classifier</td>
<td>35</td>
</tr>
<tr>
<td>2-4</td>
<td>Standard SVR and $\varepsilon$-Insensitive tube around the regression function</td>
<td>40</td>
</tr>
<tr>
<td>3-1</td>
<td>Set of pattern vectors, 3 pattern vectors in each set</td>
<td>45</td>
</tr>
<tr>
<td>3-2</td>
<td>Classification of a separable set in the presence of restricted free slack</td>
<td>47</td>
</tr>
<tr>
<td>3-3</td>
<td>Distribution of pooled free slack (PFS)</td>
<td>48</td>
</tr>
<tr>
<td>3-4</td>
<td>Distribution of free slack per set (FSS)</td>
<td>48</td>
</tr>
<tr>
<td>3-5</td>
<td>Standard SVM on a separable 3D example for different kernel bandwidths</td>
<td>54</td>
</tr>
<tr>
<td>3-6</td>
<td>Selective SVM on a separable 3D example: bold points are selected</td>
<td>55</td>
</tr>
<tr>
<td>3-7</td>
<td>Selective SVR compared to a naïve method for different bandwidths</td>
<td>63</td>
</tr>
<tr>
<td>4-1</td>
<td>Approximate placement of electrodes in the tree monkeys</td>
<td>65</td>
</tr>
<tr>
<td>4-2</td>
<td>Visual stimuli used in the experiment</td>
<td>66</td>
</tr>
<tr>
<td>4-3</td>
<td>Experimental paradigm</td>
<td>66</td>
</tr>
<tr>
<td>4-4</td>
<td>Classification and channel selection results for GE</td>
<td>77</td>
</tr>
<tr>
<td>4-5</td>
<td>Classification and channel selection results for LU</td>
<td>78</td>
</tr>
<tr>
<td>4-6</td>
<td>Classification and channel selection results for TI</td>
<td>79</td>
</tr>
<tr>
<td>4-7</td>
<td>Brain snapshots</td>
<td>80</td>
</tr>
<tr>
<td>4-8</td>
<td>a) Uncoded scores matrix b) Sorted scores c) Coded matrix d) Sorted matrix</td>
<td>81</td>
</tr>
<tr>
<td>4-9</td>
<td>a) comparative classification results b)comparative channel selection results</td>
<td>82</td>
</tr>
<tr>
<td>4-10</td>
<td>Improvement classification for line-diamond (left) and go-nogo (right)</td>
<td>83</td>
</tr>
<tr>
<td>4-11</td>
<td>Improvement in the feature selection for line-diamond (left) and go-nogo (right)</td>
<td>83</td>
</tr>
<tr>
<td>4-12</td>
<td>Baseline approach to emphasize improvement (top), difference between actual improvement and baseline (bottom)</td>
<td>84</td>
</tr>
<tr>
<td>5-1</td>
<td>Separation obtained with generalized eigenvectors</td>
<td>87</td>
</tr>
<tr>
<td>5-2</td>
<td>ReGEC algorithm</td>
<td>91</td>
</tr>
</tbody>
</table>
5-3 Separation surfaces obtained with ReGEC, GEPSVM and LIBSVM .......... 94
5-4 Classification surfaces produced by ReGEC (left) and I-ReGEC (right) .... 101
5-5 Classification consistency of I-ReGEC for random selection (left) and k-menas (right) .................................................. 103
5-6 Performance of I-ReGEC with respect to the number of starting points .... 104
NEW OPTIMIZATION METHODS AND APPLICATIONS IN KERNEL-BASED MACHINE LEARNING

By

Onur Şeref

December 2006

Chair: Panos M. Pardalos
Major Department: Industrial and Systems Engineering

In this study, new optimization methods are introduced on kernel-based machine learning. These novel methods solve real life classification problems, especially those arising in the biomedical area. The first main contribution of this study is the selective support vector machine (SelSVM) classifiers. SelSVM classifiers are motivated by the noisy temporal variations in the recordings of repeated cognitive processes, which affect the performance of standard support vector machine (SVM) classifiers. In the SelSVM classification problem there are sets of possible pattern vectors instead of individual pattern vectors. SelSVM classifiers select those pattern vectors from each set that would maximize the margin between the two classes of selected pattern vectors. SelSVM is compared with other standard alignment methods on a neural data set that is used for analyzing the integration of visual and motor cortices in the primate brain. Selective kernel-based methods are then further extended to selective support vector regression (SelSVR).

The second main contribution of this study is a fast classifier based on the standard generalized eigenvalue classifiers (GEC). The regularized GEC (ReGEC) uses a new regularization technique which reduces the solution of two eigenvalue problems in the original GEC to a single eigenvalue problem. A parallel implementation of ReGEC is developed to study large scale genomic problems. Finally, an incremental version I-ReGEC is developed to train large amounts of data efficiently. I-ReGEC incrementally builds a
substantially small subset of the training data with more consistent generalization results.
These classifiers are shown to perform comparably with the best classification methods on
publicly available benchmark classification datasets.
CHAPTER 1
INTRODUCTION

Learning is one the most distinctive innate ability of animals. A combination of experiences as input for the learning process increases the chance of survival. These experiences introduce more samples of similar conditions from which to avoid, such as the presence of a predator, or those to seek, such as finding a mate. Humans, the most advanced learners among animals, use their knowledge to create abstract models for those conditions they have not experienced, but can respond appropriately specific to an objective. Formally speaking, this abstraction can be perceived as a set of rules which can map an input condition into a set of possible responses. Intelligence can be defined as the interaction of all such rules that constitute complex cognitive tasks such as reasoning, planning, problem solving, and creating further rules for more complex scenarios with a dynamic and recursive nature.

In a world, where machines do most of the hard work, eventually, the question of making them intelligent came into picture. The initial theoretical concepts were introduced in the first half of the twentieth century, and implementations thrived with the introduction of computers, in parallel with the development of formal systems in logic and neural networks. The term artificial intelligence was coined by the beginning of the second half of the century. One of the first examples of artificial intelligence were implementations of a chess-playing program developed in 1951. Deep-Blue, a successor of such programs running on powerful computers beat the world chess champion Garry Kasparov 46 years later, in 1997. There were major accomplishments in machine learning towards the end of the century with cars equipped with AI systems which can drive for thousands of miles in traffic without a driver.

One of the major areas in AI is machine learning, which developed substantially during 1990s and 2000s, mostly as a result of the introduction of probability theory and statistical methods in AI. Machine learning can be defined as the methods developed to enable computers to learn. There have been very successful implementations ranging from
language processing to medical diagnosis and financial analysis. Most of these applications rely on *pattern recognition* which is mostly concerned with classifying objects based on their characteristics. Characteristics of an object are the qualitative and quantitative measures that distinguish it from other objects, which are also referred to as *features*. Similarity between two objects can be evaluated as a function of the differences in a set of features they possess. Based on their similarity, objects can be grouped into classes. These classes may be represented in different ways such as approximating functions or functions that define boundaries between classes. Arranging objects into such classes based on their position relative to these functions is called *classification*.

Machine learning within the classification framework can be categorized into two main classes. *Supervised learning* refers to the capability of a system to learn from a set of examples, which is a set of input/output pairs, where the input is usually a vector of features of an object, and the output is the label for the class this object belongs to. A set of objects with a feature vector and a class label is called a *training set*. This set is used to derive classification functions. The trained system is capable of predicting the label of an object. The term *supervised* originates from the fact that the labels for the objects in the training set are provided as input, and therefore were determined by an outside source, which can be considered as the *supervisor*. On the contrary, *unsupervised learning* is the case where the objects are not labeled with any class information, and learning is about forming classes of objects based on similarities between their features.

Supervised learning systems applications can be found in many fields. Financial companies prefers to classify loan requests depending on the features that characterizes loaner’s ability to pay back. Such features are learned from the historical data. A similar example is the Internal Revenue Service’s predicting tax evaders based on the features of tax evaders they detected previously. Applications may differ including cases such as a prediction system that may warn drivers about pedestrians attempting to cross the street. There are many applications in biology and medicine such as detection of cancer prone
tissues, or remote protein homology detection. Protein folding based on the DNA sequence provides important information on its expression level. More examples are available related to numerical interpolation, handwriting recognition and Montecarlo methods.

1.1 Kernel-Based Machine Learning

developed by V. Vapnik [1], Support Vector Machine (SVM) algorithms are the state-of-the-art among the classification methods in the literature. These methods classify pattern vectors which are assumed to belong to two linearly separable sets from two different classes. The classification function is defined with a hyperplane that separates both classes. Although there are infinitely many hyperplanes that separate the two classes, SVM method finds the hyperplane that maximizes the the distance from the convex hulls of both classes by solving a quadratic convex optimization problem. The success and robustness of SVM classifiers are due to their strong fundamentals on the statistical learning theory which is based on generalization bounds for SVM classifiers. These methods can be extended to the nonlinear cases by embedding the data in a nonlinear space using kernel functions [2].

SVM classifiers have been one of the most successful methods in supervised learning with applications in a wide spectrum of application areas, ranging from pattern recognition [3] and text categorization [4] to biomedicine [5–7], brain-computer interface [8, 9], and financial applications [10, 11]. The training part relies on optimization of a quadratic convex cost function subject to linear constraints. Quadratic programming (QP) is an extensively studied field of mathematics and there are many general purpose methods to solve QP problems such as quasi-newton, primal-dual and interior-point methods. The general purpose methods are suitable for small size problems. For large problems faster methods are required. These faster methods usually involve chunking [12] and decomposition [13] techniques, which use subsets of points to find the optimal hyperplane. SVM Light [14] and LIBSVM [15] are among the most frequently used implementations that use chunking and decomposition methods efficiently.
There are also alternative methods such as Generalized Proximal SVM (GEPSVM) [16] that approximate the two classes with two hyperplanes instead of separating them. The support vector machine method has been extended to perform regression. In support vector regression (SVR), class labels are replaced with scalar quantities of a dependent variable. The idea of maximizing a margin is inverted by requiring the pattern vectors to be within a specified distance from the regression function. SVR method was also developed by V. Vapnik [17], followed by some other efficient implementations [18–20].

The variety of the optimization methods and their applications have lead to the evaluation of the features that define the classification of objects. Feature selection methods were developed in order to find the most relevant features that contribute to the different classes of elements [21–24].

One of the main contributions in my dissertation is the novel concept of selective support vector machines (SelSVM). This method has $n$ sets of pattern vectors with at most $k$ pattern vectors in each set, representing $n$ single possible pattern vectors. All pattern vectors in a set share the same label. The standard SVM problem can be considered as a special case of SelSVM when $k = 1$. SelSVM picks an optimal point from each set in order to maximize the classification margin. The other pattern vectors do not have an effect on the classification function. The same idea can be applied to support vector regression with the outcome of selected vectors forming a better regression curve with less error. However, the selection process is a hard optimization problem. Therefore, relaxations of this problem are studied and shown to work very well on toy problems as well as in real life problems.

1.2 Applications to Neuroscience

The goal directed behavior develops through the activities in the integrated brain cortexes that are involved in sensory and motor processes. Understanding the structure and dynamics of this integration is one of the major goals of cognitive neuroscience. A great majority of the studies in the neurophysiology area are limited
to focusing on isolated cortical areas and their associated functions. Moreover, the
differences in the experimental designs, recording techniques and analysis methods pose
difficulties in making general inferences on the temporal coordination across cortical areas.
Simultaneous recordings of cortical activity from visual, motor and executive cortical
areas provide the essential information to explore the integration of the visual and motor
processes on a large scale.

The spatiotemporal dynamics of the visual cortical processing and the interaction
between the visual sensory cortex and motor cortex has been extensively studied in
cognitive neurophysiology. [25–31]. In recent studies, a large number of visual cortical
areas has been show to be activated rapidly as a feedforward step after the onset of
the visual stimuli [32, 33]. This mechanism serves as to prime the motor system to
prepare for the processing of the response following the determination of the stimulus
category. The ventral visual pathway is known to be involved in the visual pattern
discrimination [34, 35], where as the categorical discrimination of the stimuli is expected to
be performed in the dorsolateral prefrontal cortex [36–39]. The response related potentials
are recorded in similar studies, for even no response cases in monkeys [40, 41] and a visual
categorization task in humans [42, 43].

The majority of these studies are on visual onset latency focusing on information
processing within the visual system and ignoring the integration with the motor system.
The techniques used in the analysis of cortical activity usually involves first order
statistical measures and linear statistical models. However, it is well known that the
cortical activities and their interactions have a highly non-linear nature. Furthermore,
the ubiquity of the activities in the nervous system and the highly connected structure of
massive neural networks in the brain require methods that can incorporate simultaneous
recordings in an appropriate multidimensional domain, rather than studying individual
recordings independently.
The general purpose SVM and the feature selection methods play a critical role in my dissertation. The main data under study is the local field potentials collected from multiple channels implanted in different cortical areas of macaque monkeys while they perform a visual discrimination task that require recognition of a visual pattern followed by an appropriate response. This task is repeated to create many instances of the same experiment with different emphasis on different stimuli - response combinations. These differences are grouped as different classes of data for classification. The main objective is to be able to detect when and where these differences are observed between classes. An extensive computational effort is required to achieve robust results on a very complex, multi-dimensional and highly nonlinear data. Support vector machines are highly capable of delivering these qualities as confirmed by a large number of recent implementations. The SVM classifiers provide the temporal measure for the differentiation between the classes of different stages of the visuomotor task. In parallel with these stages, a feature selection method would provide cortical location of the differentiation for those intervals with a significant separation between different classes of recordings. The feature selection method preferred is naturally a kernel-based adaptive scaling algorithm in order to maintain consistency. This method uses support vector machines as a sub-procedure together with a combination of a conjugate gradient technique to find the relative contribution of each channel.

1.3 Regularized Generalized Eigenvalue Classifiers

Datasets in almost every application area are ever growing and are continuously updated. Moreover, numerous applications on massive datasets are emerging [44], which require efficient computational procedures to respond to the dynamics of large databases. As machine learning becomes a part of data intensive computation systems, updating the learning systems become intractable. Therefore, incremental methods that require minimal computational burden are preferred. For this purpose several methods, especially in kernel-based nonlinear classification, have been proposed to reduce the size of the training
set, and thus, the related kernel [45–49]. All of these methods show that a significant data reduction is possible while maintaining a comparable level of classification accuracy.

The binary classification problem can be formulated as a generalized eigenvalue problem [16]. This formulation differs from SVMs since, instead of finding one hyperplane that separates the two classes, it finds two hyperplanes that approximate the two classes. The prior studies require the solution of two different eigenvalue problems. The Regularized General Eigenvalue Classifier (ReGEC) is introduced, which uses a new regularization technique that requires only one eigenvalue problem to be solved. This classifier reduces the computational time by half compared to the standard eigenvalue classifiers.

Due to the size and efficiency problems, very large databases could only be processed or mined using a group of connected computers (multicomputers) that run in parallel and communicate among themselves. Standard data mining algorithms do not achieve a good performance on multicomputers, in general. Therefore, special algorithms must be designed in order to exploit their strong computational infrastructure. There are a number of comprehensive surveys on parallel implementations of widely used data mining and knowledge discovery methods and their application spectrum [50–53]. A parallel implementation of ReGEC is introduced to perform computationally hard classification tasks on genomic databases.

Another major problem in classification is the amount of training examples available. Although it looks intuitive that the more example a learning systems has, the better predictions it can make, the computational burden of learning may make the system inconvenient. One immediate solution is to select a subset of points that would retain the characteristics of the training set. A second problem arises when a new training data becomes available for training. A desirable method as a solution to the second problem should efficiently evaluate the contribution of the new data to the classification function, rather than a complete training of the incrementally augmented training set. A new method, which is referred to as Incremental ReGEC (I-ReGEC), is introduced. I-ReGEC
finds a small representative subset of the training data, which in fact, provides better
generalization results. For some publicly available benchmark classification problems,
the amount of reduction in the training set can be as large as 98% with comparable
classification accuracy and improved consistency. The proposed subset selection method
starts with an initial set of pattern vectors and incrementally expands this set by adding
new pattern vectors which contribute to improving classification accuracy. The main idea
is to use a small subset of points to solve the generalized eigenvalue problem to evaluate
the contribution of the new pattern vectors using ReGEC.

The rest of the dissertation is organized as follows. Chapter 2 discusses the
fundamental topics on kernel-based learning such as generalization, kernels, optimization
theory, support vector classification and support vector regression. In Chapter 3, these
classification and regression problems are further developed into selective classification
and regression problems that are intended to find those pattern vectors which provide
a better separation of two classes of data, or vectors that define a regression function
with lower error. In Chapter 4, the standard support vector classification, regression and
feature selection methods are applied to neural data from a visuomotor task performed
by macaque monkeys in order to study the integration of visual and motor systems in
the primate brain. The results are further improved by applying selective support vector
classification. In Chapter 5, extensions to generalized eigenvalue classifiers are presented
involving a new regularization technique, a parallel implementation of the regularized
classifier, and a fast incremental classification algorithm that uses the regularized classifier
to find a small subset of the training set with better generalization capabilities and
comparable classification accuracy. Finally, the conclusions regarding the kernel-based
methods and their applications are drawn in Chapter 6.
CHAPTER 2
KERNEL-BASED LEARNING METHODS

This chapter covers the fundamental topics on machine learning, specifically kernel methods in machine learning, which are built on the strong theoretical background of the statistical learning theory [17]. Although kernel methods are relatively new topics in machine learning there are comprehensive books available on kernel machine learning [54, 55] and most of the material presented in this chapter are compiled from these books.

The sequence of topics covered in this chapter are as follows. First, the concept of kernels is explained and how they are used as nonlinear mapping tools in order to enhance similarity among data points is shown. Then, the generalization theory is reviewed briefly to mention bounds of the empirical risk based on the Vapnik-Chervonenkis theory. Some fundamental concepts in optimization, more specifically, some elements of the Lagrangian theory are introduced to explain necessary and sufficient conditions for convex optimization problems with linear inequality constraints. These three topics pave the way to support vector classification and regression models with different variations of primal and dual formulations with respect to penalization schemes on the classification error.

2.1 Kernels

Kernels are introduced in classification to provide enhanced similarity measures between pattern vectors. They basically transform the, so called, input space, \( \mathcal{X} \), in which the original pattern vectors reside, to a usually higher dimensional dot-product space \( \mathcal{H} \) called the feature space, via a map \( \Phi : \mathcal{X} \rightarrow \mathcal{H} \), such that

\[ K(x_i, x_j) = \langle \Phi(x_i) \cdot \Phi(x_j) \rangle. \] (2-1)

The main concept is focused on the dot product of two mapped pattern vectors. The dot product of mapped pattern vectors may become computationally intractable, while
finding the dot product implicitly has same complexity as in the linear case, in general. As an example, consider the following dot product in 2-D. [55] (pg. 26)

\[
\Phi : \mathbb{R}^2 \rightarrow \mathcal{H} = \mathbb{R}^3 \\
(x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2),
\]

where \(x_{(i)}\) is the \(i^{th}\) component of the pattern vector \(x\). Note that for two pattern vectors \(x_i\) and \(x_j\), \(\langle \Phi(x_i) \cdot \Phi(x_j) \rangle = (\langle x_i \cdot x_j \rangle)^2\). A visualization of this mapping is given on a small toy example in Figure 2-1. Another mapping from 2-D input space to a 4-D feature for the same kernel is as follows

![Figure 2-1. Kernel mapping of circular data in 2-D to 3-D.](image-url)
\(\Phi : \mathbb{R}^2 \rightarrow \mathcal{H} = \mathbb{R}^4\)

\[(x_1, x_2) \mapsto (x_1^2, x_2^2, x_1x_2, x_1x_2),\]

The following proposition demonstrates a generalization on polynomial approximation for any dimensional input vectors, and any degree of polynomial kernel [56].

**Proposition 2.1.1.** Define \(\Phi_k\) to map \(x \in \mathbb{R}^d\) to the vector \(\Phi_k(x)\) whose entries are all possible \(k^{th}\) degree ordered products of the entries of \(x\). Then, the corresponding kernel computing the dot product of vectors mapped by \(\Phi_k\) is

\[K(x_i, x_j) = \langle \Phi_k(x_i) \cdot \Phi_k(x_j) \rangle = \langle x_i \cdot x_j \rangle^k. \quad (2-2)\]

The results on unordered product features are also given in [55].

Now we can investigate the conditions how kernels can create feature maps. More generally, given a kernel, how can we construct a feature space such that the kernel computes the dot product in that kernel space? First we define a positive semidefinite kernel.

**Definition 2.1.2 (Positive definite Kernel).** Let \(\mathcal{X}\) be a nonempty set. A function \(K\) on \(\mathcal{X} \times \mathcal{X}\) which for all \(x_1, \ldots, x_n \in \mathcal{X}\) gives rise to a positive definite matrix \(K\) such that

\[\sum_{i,j} c_i c_j K_{ij} \geq 0\]

for all \(c_i \in \mathbb{R}\), is called a positive definite kernel.

In general, if we relax the semi-definite positivity condition on \(K\), the resulting matrix is called Gram matrix or kernel matrix. The term kernel stems from the first use of integral operators, studied by the famous mathematician David Hilbert for functions \((T_k f)\) that give rise to an operator \(T_k\) such that

\[(T_k f)(x) = \int_{\mathcal{X}} K(x, x^*) f(x^*) dx^*. \quad (2-3)\]

The function \((2-3)\) above is called the kernel of \(T_k\).
It is worth noting that a positive definite kernel implies positivity on the diagonal and symmetry. That is,

\[
K(x, x) \geq 0 \quad \text{for all } x \in \mathcal{X},
\]

\[
K(x_i, x_j) = K(x_j, x_i).
\] (2–4)

In the literature it was shown that any algorithm that works on dot products can be kernelized through the \textit{kernel trick}, as explained below:

\textbf{Remark 2.1.3 (Kernel Trick [55] (pg. 34)). Given an algorithm which is formulated in terms of a positive definite kernel } K \textit{, one can construct an alternative algorithm by replacing } K \textit{ by another positive definite kernel } \tilde{K} \textit{.}

In the machine learning literature, this kernel trick is introduced by Mercer’s theorem and explains the geometry of feature spaces. It can be considered as the characterization of a kernel function \( K(x, bf \cdot) \).

\textbf{Theorem 2.1.4 (Mercer [54] (pg. 35)). Let } \mathcal{X} \textit{ be a compact subset of } \mathbb{R}^n. \textit{Suppose } K \textit{ is a continuous symmetric function such that the integral operator } T_K : L^2(\mathcal{X}) \rightarrow L^2(\mathcal{X}),

\[
(T_k f)(\cdot) = \int_{\mathcal{X}} K(\cdot, x) f(x) dx
\] (2–5)

\textit{is positive, that is}

\[
\int_{\mathcal{X} \times \mathcal{X}} K(x^*, x) f(x^*) f(x) dx^* dx \geq 0,
\] (2–6)

\textit{for all } f \in L^2(\mathcal{X}). \textit{Then we can expand } K(x^*, x) \textit{ in a uniformly convergent series in terms of } T_K \textit{'s eigenfunctions, } phi_j \in L^2(\mathcal{X}), \textit{normalized in such a way that } \|\phi_j\|_{L^2} = 1, \textit{and positive associated eigenvalues } \lambda_j \geq 0,

\[
K(x^*, x) = \sum_{j=1}^{\lambda} \phi_j(x^*) \phi(x).
\] (2–7)
The conditions for Mercer’s theorem are equivalent to the requirement that the corresponding matrix is positive semi-definite for any finite subset of $X$. The term kernel will be used for the rest of the text to refer to those functions that satisfy this property, although it is not the general definition of a kernel. This concludes the discussion on kernels to be able to develop kernel-based methods within the intended scope of the study.

### 2.2 Generalization Theory

In this section, we review the factors that affect good generalization results and give an overview of the Vapnik-Chervonenkis (VC) theory which provides robust bounds on linear classifiers in kernel spaces. The results of this theorem are quite fundamental to machine learning since the flexibility introduced with kernels would lead to overfitting, which is contrary to the generalization of the classification function. The generalization bounds for maximal margin classifiers, soft margin classifiers and generalization for regression are discussed briefly.

The data used in training and testing are assumed to be generated from the same unknown distribution $D$ over the pairs $(x_i, y_i)$. The probably approximately correct (PAC) bound on learning requires a strong bound that is not likely to fail. PAC has the form $\varepsilon = \varepsilon(n, H, \delta)$, which means that, with probability of $1 - \delta$ over randomly generated sets $S$, the generalization error of selected hypothesis $h_S$ will be bounded by

$$\text{err}_D(h_S) \leq \varepsilon(n, H, \delta),$$

(2–8)

where $H$ is the class of all possible hypothesis and $n$ is the number of examples in the training set. Before presenting further results, we mention that the quantity known as Vapnik-Chervonenkis (VC) dimension shows the flexibility or capacity of a function class to classify a random training set of points. Here, we give the fundamental theorem of learning, details of which can be found in [54].

**Theorem 2.2.1** (Vapnik-Chervonenkis). Let $H$ be a hypothesis space having a VC dimension of $d$. For any probability distribution on $X \times \{-1, 1\}$, with probability $1 - \delta$ over
Theorem 2.2.2 (Error bound on maximal margin classifier). Consider thresholding real values linear functions $L$ with unit weight vectors on an inner product space $X$ and fix $\gamma \in \mathbb{R}^+$. For any probability distribution $D$ on $X \times \{-1,1\}$ with support in a ball of radius $R$ around the origin, with probability $1 - \delta$ over $n$ random examples $S$, any hypothesis $f \in L$ that has margin $m_S(f) \geq \gamma$ on $S$ has error no more than,

$$
\text{err}_D \leq \varepsilon(n, L, \delta, \gamma) = \frac{2}{n} \left( 64R^2\gamma^2 \log \frac{en\gamma}{4R} \log \frac{128nR^2}{\gamma^2} + \log \frac{4}{\delta} \right),
$$

provided $n > 2/\varepsilon$ and $64R^2/\gamma^2 < n$.

Theorem 2.2.3 (Error bound on soft margin classifier). Consider thresholding real-valued linear functions $L$ with unit weight vectors on an inner product space $X$ and fix $\gamma \in \mathbb{R}^+$. There is a constant $c$, such that for any probability distribution $D$ on $X \times \{-1,1\}$ with support in a ball of radius $R$ around the origin, with probability $1 - \delta$ over $n$ random examples $S$, any hypothesis $f \in L$ has no more than

$$
\text{err}_D \leq \frac{c}{n} \left( \frac{R^2 + \|\xi\|_2^2}{\gamma^2} \log^2 n + \log \frac{1}{\delta} \right),
$$

where $\xi = \xi(f, S, \gamma)$ is the margin slack vector with respect to $f$ and $\gamma$.

The generalization error bound for the 1-norm soft margin classification is very similar to that of the 2-norm with an additional term $\log(1/\gamma)$ in the numerator of the first term in parenthesis, which is multiplied by $\|\xi\|_2^2$. 

\[err_D \leq \frac{c}{n} \left( \frac{R^2 + \|\xi\|_2^2}{\gamma^2} \log^2 n + \log \frac{1}{\delta} \right), \]
Theorem 2.2.4 (Error bound on regression). Consider performing regression with linear functions $L$ on an inner product space $X$ and fix $\gamma \leq \theta \in \mathbb{R}^+$. There is a constant $c$, such that for any probability distribution $D$ on $X \times \mathbb{R}$ with support in a ball of radius $R$ around the origin, with probability $1 - \delta$ over $n$ random examples $S$, the probability that a hypothesis $w \in L$ has output more than $\theta$ away from its true value is bounded by,

$$\text{err}_D \leq \frac{c}{n} \left( \frac{\|w\|^2 + \|\xi\|^2 \log(1/\gamma)}{\gamma^2} \log^2 n + \log \frac{1}{\delta} \right),$$

(2–12)

where $\xi = \xi(w, S, \theta, \gamma)$ is the margin slack vector with respect to $w$, $\theta$, and $\gamma$.

These bounds provide good generalization results on the support vector machine classification and regression methods. The only remaining topic, before the introduction of SVM methods is optimization theory which explains the conditions under which we can solve the problem of finding a hyperplane that maximizes the margin, or a regression function that minimizes the error.

### 2.3 Optimization Theory

Optimization theory plays an important role in machine learning. The machine learning problems can be formulated as optimization problems, for which well studied efficient solution methods are known. The mathematical framework provides necessary and sufficient conditions for the optimum solution. These conditions, in turn, help determine the desired classification function that provide a good generalization on the data. The primal optimization problem is presented in a general form, which has an associated dual problem. As it will be explained in Section 2.4.1 the latter formulation provides the means for the kernel methods to be applicable in support vector classifiers. The transformation from the primal problem to the dual problem is explained through Lagrangian theory, which is also a well studied subject in mathematics. We start by the definition of the primal optimization problem.

**Definition 2.3.1** (Primal problem). Given functions $f$, $g_i$, $i=1, \ldots, k$, and $h_i$, $i=1, \ldots, m$, defined on a domain $\Omega \subseteq \mathbb{R}^d$,
\[
\begin{align*}
\text{minimize} & \quad f(w), & w \in \Omega, \\
\text{subject to} & \quad g_i(w) \leq 0, & i = 1, \ldots, k, \\
& \quad h_i(w) = 0, & i = 1, \ldots, m,
\end{align*}
\]

(2–13) (2–14) (2–15) (2–16)

The optimization problem that is rendered by the support vector classification is a convex function on a convex set. First we define the convex function and convex set.

**Definition 2.3.2 (Convex function).** A real valued function \( f(w) \) is called convex for \( w \in \mathbb{R}^d \) if for \( w, u \in \mathbb{R}^d \), and for any \( \theta \) in \([0, 1]\),

\[
f(\theta w + (1 - \theta)u) \leq \theta f(w) + (1 - \theta) f(u).
\]

(2–17)

Continuous functions that are twice differentiable are also convex if their Hessian matrix is positive semi-definite.

**Definition 2.3.3 (Convex set).** A set is convex if for any \( w, u \in \Omega \), the point

\[
r = (\theta w + (1 - \theta)u) \in \Omega \quad \text{for all} \ \theta \in [0, 1].
\]

Now, we restrict the optimization problem such that the objective function is quadratic, the constraints are linear, and the domain \( \Omega = \mathbb{R}^n \). Next, we review the Lagrangian theory and Karush-Kuhn-Tucker (KKT) optimality conditions for the dual transformation.

Lagrangian theory is based on characterizing an optimization problem without inequality constraints. The *Lagrangian function* and the *Lagrangian multipliers* are the main concepts in this theory. Karush-Kuhn-Tucker, further improved this characterization to define optimality conditions when the problem has inequality constraints. We give formal definitions of these concepts below. We start with the most general case of
the Lagrangian function, where the optimization function includes both equality and inequality constraints.

**Definition 2.3.4** (Lagrangian function and multipliers). The Lagrangian function of an optimization problem with domain $\Omega \in \mathbb{R}^d$,

\[
\text{minimize} \quad f(w) \quad w \in \Omega \\
\text{subject to} \quad g_i(w) \leq 0 \quad i = 1, \ldots, k, \\
\quad h_i(w) = 0 \quad i = 1, \ldots, m,
\]

is defined as

\[
L(w, \alpha, \beta) = f(w) + \sum_{i=1}^{m} \alpha_i g_i(w) + \sum_{i=1}^{m} \beta_i h_i(w),
\]

and the coefficients $\alpha_i$ and $\beta_i$ are called the Lagrangian multipliers.

Note that the Lagrangian function is a relaxation of the original optimization problem (2–18), and sets a lower bound on the optimal solution value of the original problem.

Before we show this relation, let us define the Lagrangian dual formulation.

**Definition 2.3.5** (Lagrangian dual problem). The lagrangian dual problem of the problem (2–18) is defined as follows.

\[
\max_{\alpha, \beta} \quad \theta(\alpha, \beta) = \{ \inf_{w \in \Omega} L(w, \alpha, \beta) \}
\]

\[
\text{subject to} \quad \alpha \geq 0.
\]

The following theorem establishes the relationship between the solution to the primal problem and the dual problem.

**Theorem 2.3.6.** Let $w \in \Omega$ be a feasible solution of the primal problem and $(\alpha, \beta)$ a feasible solution of the dual problem. Then $f(w) \geq \theta(\alpha, \beta)$. 

28
Proof. By definition and the feasibility of $w, \alpha$ and $\beta$,

$$
\theta(\alpha, \beta) = \inf_{u \in \Omega} L(u, \alpha, \alpha) \\
\leq L(w, \alpha, \beta) \\
= f(w) + \alpha^T g(w) + \beta^T h(w) \leq f(w).
$$

A useful corollary follow, basically stating that the optimality of the solutions to the primal and dual problems can be concluded when the objective functions of two problems are equal.

Corollary 2.3.7. If $f(w^*) = \theta(\alpha^*, \beta^*)$, and $w$, and $(\alpha, \beta)$ are feasible for the primal and dual problems, then they are optimal solutions to the primal and dual problems, respectively, in which case $\alpha_i^* g_i(w^*) = 0$.

Proof. The results follow from the inequalities in the proof of Theorem 2.3.6. \qed

The solutions to the primal and dual problems are not guaranteed to be equal, in which case the difference is referred as the duality gap. The following theorem states the strong duality when certain conditions are satisfied for an optimization problem.

Theorem 2.3.8 (Strong duality). Given an optimization problem with convex domain $\Omega \subseteq \mathbb{R}^d$,

$$
\min_{w \in \Omega} f(w) \quad \text{(2–22)}
$$

subject to $g_i(w) \leq 0$, $i = 1, \ldots, k$,

$h_i(w) = 0$, $i = 1, \ldots, m$,

where $g_i$ and $h_i$ are affine functions, i.e. $g(w) = Aw - b$, for some matrix $A$ and vector $b$, the duality gap is zero.

This theorem is followed by the Kuhn-Tucker theorem, which states the conditions such that the results of strong duality theorem (Theorem 2.3.8) can be applied in practice.
Theorem 2.3.9. Given an optimization problem as in Theorem 2.3.8, where \( f(w) \) is convex, the necessary and sufficient conditions for a point \( w^* \) to be optimum are the existence of \( \alpha \) and \( \beta \) such that

\[
\frac{\partial L(w, \alpha, \beta)}{\partial w} = 0, \\
\frac{\partial L(w, \alpha, \beta)}{\partial \beta} = 0, \\
\alpha^*_i g_i(w^*) = 0, \quad i = 1, \ldots, k, \\
g_i(w) \leq 0 \quad i = 1, \ldots, k, \\
\alpha^*_i \geq 0 \quad i = 1, \ldots, k.
\]

The relation \( \alpha_i g_i(w) = 0 \) is known as the Karush-Kuhn-Tucker complementarity condition, and implies that \( \alpha_i \geq 0 \) for active constraints, and \( \alpha_i = 0 \) for inactive constraints. This means that small perturbations of the inactive constraints have no effect on the optimal solution.

The main convenience of the Kuhn-Tucker theorem is transforming the primal problem into a dual problem which is easier to solve. This is mainly done by differentiating the Lagrangian function with respect to the primal variables, and substituting the equivalent expressions back in the Lagrangian function to achieve a simpler formulation without inequality constraints. The solution to the dual problem shows which constraints in the primal problem are active. The active constraints indicate support vectors, which are generally much less in number compared to the total number of all inequality constraints. In machine learning, this result will become more clear in the next section.

2.4 Support Vector Machines

In this section, we coalesce all of the results we reviewed before into building efficient and effective classification methods. Namely, the dot products and kernel spaces provide the foundation for linear and non-linear mapping of the pattern vectors, where as generalization theory explains the classification error bounds, and finally, optimization
theory introduces the tools and techniques for efficient implementation of the nonlinear methods. We first discuss the *Maximal Margin Classifier*, which is the simplest form of SVM that work for separable classes. Next, we allow some misclassification together some penalty in *Soft Margin Classifiers*.

### 2.4.1 Maximal Margin Classifier

We introduce maximal margin classifier as the simplest form of SVM classification. The underlying optimization problem is only feasible for separable cases, which does not fit well with most of the real life classification problems. However, the maximal margin classifier demonstrates the fundamental technique used in the more commonly used versions of SVM classifiers. The key factor is that the solution to the optimization problem minimizes the bound on the generalization error by maximizing the margin with respect to the training set. Since this bound does not depend on the dimensionality of the space the same method can be used in a kernel induced space. The basic premise of this method lies in the minimization of a convex optimization problem with linear inequality constraints, which can efficiently be implemented based on the well-studied optimization theory we have reviewed in Section 2.3.

First we introduce the problem by formal definitions of a hyperplane, a canonical hyperplane, and the margin of a hyperplane. A hyperplane is a $d-1$ dimensional linear surface in a $d$ dimensional space $\mathcal{H}$, and it can be represented as:

$$\{ x \in \mathcal{H} | \langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0 \}, \quad \mathbf{w} \in \mathcal{H}, \; b \in \mathbb{R}. \quad (2-23)$$

In $2-23$, $\mathbf{w}$ is orthogonal to the hyperplane and $\langle x \cdot \rangle$ gives the length of pattern vector $x$ along the direction $\mathbf{w}$ when $\mathbf{w}$ is a unit vector. Note that we have inherent degree of freedom in specifying the same hyperplane as $(\lambda \mathbf{w}, \lambda b)$ for $\lambda \neq 0$.

**Definition 2.4.1 (Canonical Hyperplane).** A hyperplane $(\mathbf{w}, b)$ is a called a canonical hyperplane if the closest point from the hyperplane has a distance $1/\|\mathbf{w}\|$, i.e.

$$\min_{i=1,...,m} |\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b| = 1.$$
Now, consider two points, each belonging to one class, $x^-$ and $x^+$. We want to find a canonical hyperplane $(w, b)$ such that

\[
\langle w \cdot x^+ \rangle + b = 1 \quad (2-24)
\]
\[
\langle w \cdot x^- \rangle + b = -1. \quad (2-25)
\]

This implies a margin of

\[
\frac{\langle w/\|w\| \cdot (x^+ - x^-) \rangle}{2} = 1/\|w\|. \quad (2-27)
\]

We assume that the training data and the test data come from the same distribution. Due to the noise in the data, the separating hyperplane can still classify new pattern vectors correctly if the margin is maximized. Maximizing the margin $1/\|w\|$ for the canonical hyperplane is equivalent to minimizing $\|w\|$. In the following optimization problem, each point $x_i$ with a label $y_i$ satisfies the canonicality condition in the constraints 2–28, while maximizing the margin by minimizing the norm of the normal vector $w$ of the hyperplane, as shown in Figure 2-2.
$\min \quad \frac{1}{2}\|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i^2$  \hfill (2–28) \\
subject to 
\[ y_i(\langle w \cdot x_i \rangle + b) \geq 1 - \xi_i \]  \hfill (2–29)

Problem 2–28 is referred as the primal problem. In order to apply kernel methods, the dual problem can be derived from the Lagrangian function 2–30.

$$L(w, b, \xi, \hat{\xi}) = \frac{1}{2}\|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i^2$$  \hfill (2–30)
$$- \sum_{i=1}^{n} \alpha_i [y_i(\langle w \cdot x_i \rangle + b) - 1 + \xi_i]$$  \hfill (2–31)

Differentiating $L(w, b, \alpha)$ with respect to the primal variables $w$ and $b$, and assuming stationarity, we get the following expressions, which we can substituted back in the Lagrangian function to obtain the dual formulation with only dual variables.

$$\frac{\partial L(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{n} y_i \alpha_i x_i = 0,$$
$$\frac{\partial L(w, b, \alpha)}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0. \hfill (2–32)$$

The following proposition shows the dual formulation and how the solution to the dual formulation realizes the maximal hyperplane.

**Proposition 2.4.2.** Consider a linearly separable training sample $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and suppose $\alpha^*$ solve the following quadratic optimization problem.
\[
\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle,
\]

subject to \[\sum_{i=1}^{n} y_i \alpha_i = 0,\]
\[\alpha \geq 0.\]

Then \(\mathbf{w}^* = \sum_{i=1}^{n} y_i \alpha_i^* \mathbf{x}_i\) realizes the maximal margin hyperplane with geometric margin \(\gamma = \frac{1}{||\mathbf{w}^*||}\).

Note that from Karush-Kuhn-Tucker complementarity conditions, the corresponding constraints in the primal problem are active for those with the dual variable \(\alpha_i^* = 0\).

Knowing \(\mathbf{w}^*\), we can find \(b^*\) as follows:

\[
b^* = \sum_{i: \alpha_i^* > 0} y_i - \langle \mathbf{w}^* \cdot \mathbf{x}_i \rangle.
\]

Note that the distance of a pattern vector to the hyperplane is given by the function

\[
f(\mathbf{x}, \alpha^*, b^*) = \sum_{i=1}^{n} y_i \alpha_i^* \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b.
\]

since the distance on either side of the hyperplane would have opposite signs, the following function can classify a pattern vector \(\mathbf{x}\) at the right side of the hyperplane.

\[
\text{class}(\mathbf{x}) = \text{sgn}\{f(\mathbf{x}, \alpha^*, b^*)\}.
\]

This classification method can easily be extended to the feature space by replacing the linear dot product \(\langle x_i \cdot x_j \rangle\) with a kernel \(K(\mathbf{x}_i, \mathbf{x}_j)\). Since the linear dot product is a linear kernel, we use a generic kernel in further formulations.

2.4.2 Soft Margin Classifier

Most real life problems have non separable data, even in the feature space. This is caused by the noise in the data. However, the maximal margin classifier can find a
separation in the feature space with the help of suitable kernels. This usually results in overfitting. Slack variables can tolerate the noise in the data well, and better generalization results can be achieved. The slack variables $\xi_i$ allow misclassification for each pattern vector if needed. However, the slack variables, when greater than zero, are subject to a penalty of $C$. In Figure 2-3 soft margin is demonstrated which incurs penalty for misclassified pattern vectors. Now, we can modify the maximum margin classifier with the introduction of slack variables, as follows.

$$\min \frac{1}{2}\|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i^2$$
subject to \( y_i(\langle w \cdot x_i \rangle + b) \geq 1 - \xi_i \)

Note that $\xi_i < 0$ cannot be true for any optimal solution to 2–39, since the corresponding constraint would still be satisfied while incurring less cost if $\xi_i = 0$. Therefore, we do not explicitly need a positivity condition on the slack variables.

Figure 2-3. Soft margin classifier
In 2–39 the 2-norm of the slack are penalized in the objective. An alternative formulation 2–39 involves penalization of the 1-norm slack variables in the objective. However, we need to impose positivity on the slack variables be able to penalize them.

\[
\min \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i
\]

subject to \( y_i((w \cdot x_i) + b) \geq 1 - \xi_i \)
\[
\xi_i \geq 0 \quad i = 1, \ldots, n.
\]

Now we can derive the dual formulation for both 1-norm and 2-norm support vector machine formulations. The significance of these formulations are that they do not involve inequality constraints, and they allow kernels to be introduced in the objective function where a dot product occurs. The standard method is to differentiate the Lagrangian function with respect to the primal variables and substitute them back in the Lagrangian function. The Lagrangian function for the 2-norm SVM primal problem is as follows.

\[
L(w, b, \xi, \alpha) = \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i^2
\]

\[
- \sum_{i=1}^{n} \alpha_i [y_i((w \cdot x_i) + b) - 1 + \xi_i]
\]

Differentiating the Lagrangian function with respect to \(w\), \(b\), and each \(\xi\), we get

\[
\frac{\partial L(w, b, \xi, \alpha)}{\partial w} = w - \sum_{i=1}^{n} y_i \alpha_i x_i = 0
\]

\[
\frac{\partial L(w, b, \xi, \alpha)}{\partial b} = \sum_{i=1}^{n} y_i a_i = 0
\]

\[
\frac{\partial L(w, b, \xi, \alpha)}{\partial \xi_i} = C \xi_i - \alpha_i = 0.
\]
Substituting the partial derivatives \((2-43-2-45)\) in \((2-41)\), we obtain,

\[
\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i \cdot x_j \rangle - \frac{1}{2C} \sum_{i=1}^{n} \alpha_i^2
\]

\[(2-46)\]

We can substitute the dot product in the objective with a more general kernel of choice. The following proposition summarizes the dual formulation for 2-norm soft margin SVM.

**Proposition 2.4.3.** Consider classifying a training sample \(S = \{(x_1, y_1), \ldots, (x_n, y_n)\}\) using a feature space induced by a kernel \(K(\cdot, \cdot)\), and suppose \(\alpha^*\) is a solution to the following problem:

\[
\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \frac{1}{2C} \sum_{i=1}^{n} \alpha_i^2
\]

\[(2-47)\]

subject to \(\sum_{i=1}^{n} y_i \alpha_i = 0,\)

\(\alpha_i \geq 0 \quad i = 1, \ldots, n.\)

Then, the decision rule \(\text{sgn}(f(x))\) is equivalent to the hyperplane in the feature space induced by \(K(\cdot, \cdot)\), where

\[
f(x) = \sum_{i=1}^{n} y_i \alpha_i^* K(x, x_i) + b^*
\]

\[(2-48)\]

and \(b^*\) is chosen such that \(y_i f(x_i) = 1 - \alpha_i^*/C\) for those pattern vectors with \(\alpha_i^* > 0\).

The derivation for the 1-norm dual formulation is very similar to that of 2-norm. The Lagrangian function is slightly different, with the sum of slack variables, and the additional term for the positivity constraints.
\[ L(w, b, \xi, \alpha, r) = \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i \]

\[ - \sum_{i=1}^{n} \alpha_i [y_i(\langle w \cdot x_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{n} r_i \xi_i. \]

Differentiating the lagrangian function with respect to \( w, b \) and \( \xi \), and imposing stationarity, we obtain,

\[ \frac{\partial L(w, b, \xi, \alpha)}{\partial w} = w - \sum_{i=1}^{n} y_i \alpha_i x_i = 0 \]

(2–51)

\[ \frac{\partial L(w, b, \xi, \alpha)}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0 \]

(2–52)

\[ \frac{\partial L(w, b, \xi, \alpha)}{\partial \xi_i} = C - \alpha_i - r_i = 0. \]

(2–53)

Substituting these expressions back in 2–49, we obtain,

\[ \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i \cdot x_j \rangle \]

The only difference between the 1-norm and 2-norm are that from 2–53 and \( r_i \geq 0 \), the dual variables \( \alpha_i \) should be in \([0, C]\), which is often referred to as the box constraint. The following proposition summarizes the result for 1-norm dual formulation for classification.

**Proposition 2.4.4.** Consider classifying a training sample \( S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \) using a feature space induced by a kernel \( K(\cdot, \cdot) \), and suppose \( \alpha^* \) is a solution to the following problem:

\[ \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_l) \]

(2–54)

subject to \( \sum_{i=1}^{n} y_i \alpha_i = 0 \),

\[ 0 \leq \alpha_i \leq C \quad i = 1, \ldots, n. \]
Then, the decision rule \( \text{sgn}(f(x)) \) is equivalent to the hyperplane in the feature space induced by \( K(\cdot, \cdot) \), where

\[
f(x) = \sum_{i=1}^{n} y_i \alpha_i^* K(x, x_i) + b^*
\]

and \( b^* \) is chosen such that \( y_i f(x_i) = 1 \) for those pattern vectors with \( 0 < \alpha_i^* < C \).

### 2.5 Gradient Based Adaptive Scaling

This approach introduces a global optimization problem and finds certain parameters regarding the separating hyperplane and linear scaling factors for the input space.

This linear scaling consist in determining a diagonal matrix \( \Sigma \), where each element on the vector of diagonal entries \( \sigma \) is the weight of the corresponding feature. The purpose of the scaling is to obtain a linear transformation of the input space that improves the classification rate. In this method the following global optimization problem is defined:

\[
\max_{\sigma, \xi, \alpha} \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j K(\hat{x}_i, \hat{x}_j) + C \sum_{i=1}^{\ell} \xi_i
\]

\[
s.t. \quad y_i \left( \sum_{j=1}^{\ell} \alpha_j y_j K(\hat{x}_i, \hat{x}_j) \right) \geq 1 - \xi_i \quad i = 1, \ldots, \ell
\]

\[
\xi_i \geq 0 \quad i = 1, \ldots, \ell
\]

\[
\frac{1}{n} \sum_{k=1}^{n} \sigma_k^p = \sigma_0^p
\]

\[
\sigma_k \geq 0 \quad k = 1, \ldots, n
\]

where \( \hat{x}_i = \Sigma x_i \), and the parameter \( p \) has the role to encourage the sparsity of the solution for \( \Sigma \).

However, this problem is hard to solve. Therefore, an iterative two stage algorithm is proposed [22]. In the first stage of any iteration \( i \) with the starting scale vector \( \sigma(i - 1) \), the problem is optimized with respect to vector \( \alpha \) as in the standard SVM method to find a solution \( \alpha(i) \). In the second stage, the problem is solved with respect to the scale vector.
σ to find a steepest ascend σ(i). However α(i) are fixed to simplify the problem assuming that α(i + 1) will not be affected by σ(i) substantially.

2.6 Support Vector Regression

The structure of the support vector regression is very similar to the classification method. Basically, we try to estimate a linear function in a kernel induced nonlinear space. The objective is to minimize the generalization bounds while ignoring errors within a given value \( \varepsilon \). This can be thought as a hyper-tube around a linear function in the kernel induced feature space, such that the pattern vectors that are in this tube are assumed not to contribute any error as in Figure 2-4.

![Figure 2-4. Standard SVR and \( \varepsilon \)-Insensitive tube around the regression function](image)

Since anything in the \( \varepsilon \) of the estimated regression function does not contribute as error, this type of regression is called \( \varepsilon \)-Insensitive. As in the classification case, there are two versions of the \( \varepsilon \) defined by the norm of the error, which are called Quadratic and Linear \( \varepsilon \)-Insensitive support vector regression, respectively.

2.6.1 Quadratic \( \varepsilon \)-Sensitive Loss

The main idea is to create a linear function in the kernel induced space such that the loss function for regression from the generalization theory is minimized. The corresponding primal problem is given as in (2–57). The constraints imply that the actual pattern
vectors are allowed to be $\varepsilon$ above (constraint 2–58) or below (constraint 2–59) the regression function without penalty. All pattern vectors outside the $\varepsilon$ range are still allowed, however they incur a cost of $C$.

$$\min \quad \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} (\xi_i^2 + \hat{\xi}_i^2)$$  \hspace{1cm} (2–57)

subject to \hspace{1cm} 
\begin{align*}
(\langle w \cdot x_i \rangle + b) - y_i &\leq \varepsilon + \xi_i \hspace{1cm} (2–58) \\
y_i - (\langle w \cdot x_i \rangle + b) &\leq \varepsilon + \hat{\xi}_i. \hspace{1cm} (2–59)
\end{align*}

In order to apply the kernel trick, we need to derive the dual formulation from the Lagrangian function for (2–57), which is given as,

$$L(w, b, \xi, \hat{\xi}) = \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} (\xi_i^2 + \hat{\xi}_i^2)$$  \hspace{1cm} (2–60)

\begin{align*}
- \sum_{i=1}^{n} \alpha_i (\varepsilon + \xi_i + y_i - \langle w \cdot x_i \rangle - b)
- \sum_{i=1}^{n} \hat{\alpha}_i (\varepsilon + \hat{\xi}_i - y_i + \langle w \cdot x_i \rangle + b)
\end{align*}

Differentiating the Lagrangian function with respect to the primal variables $w$, $b$, and each $\xi$ and $\hat{\xi}$, we get

$$\partial L/\partial w = w - \sum_{i=1}^{n} (\hat{\alpha}_i - \alpha_i) x_i = 0$$  \hspace{1cm} (2–61)

$$\partial L/\partial b = \sum_{i=1}^{n} (\alpha_i - \hat{\alpha}_i) = 0$$  \hspace{1cm} (2–62)

$$\partial L/\partial \xi_i = C \xi_i - \alpha_i = 0$$  \hspace{1cm} (2–63)

$$\partial L/\partial \hat{\xi}_i = C \hat{\xi}_i - \hat{\alpha}_i = 0$$  \hspace{1cm} (2–64)

Substituting the partial derivatives (2–61 - 2–64) in (2–60), we can write the following dual formulation,
\[
\max -\frac{1}{2C} \sum_{i=1}^{n} (\alpha_i^2 + \hat{\alpha}_i^2) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_i - \alpha_i)\langle x_i \cdot x_j \rangle \tag{2–65}
\]
\[
-\varepsilon \sum_{i=1}^{n} (\alpha_i + \hat{\alpha}_i) + \sum_{i=1}^{n} y_i(\hat{\alpha}_i - \alpha_i)
\]
subject to
\[
\sum_{i=1}^{n} (\alpha_i - \hat{\alpha}_i) = 0
\]
\[
\alpha_i, \hat{\alpha}_i \geq 0 \quad \text{for } i = 1, \ldots, n.
\]

From the solution \(\alpha^*\) and \(\hat{\alpha}^*\), the resulting regression function is given as,

\[
f(w) = \sum_{i=1}^{n} (\hat{\alpha}_i^* - \alpha_i^*)K(x, x_i) + b^*, \tag{2–66}
\]
where, \(b^*\) is chosen such that \(f(x_i) - y_i = -\varepsilon - (\hat{\alpha}_i^* - \alpha_i^*)/C\) for any \(i\) with \((\hat{\alpha}_i^* - \alpha_i^*) > 0\).

### 2.6.2 Linear \(\varepsilon\)-Sensitive Loss

The linear \(\varepsilon\)-sensitive loss version is slightly different from the quadratic loss formulation, where we impose positivity on the slack variables while penalizing them in the objective function. The formulation is given as follows,

\[
\min : \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} (\xi_i + \hat{\xi}_i) \tag{2–67}
\]
subject to
\[
(\langle w \cdot x_i \rangle + b) - y_i \leq \varepsilon + \xi_i
\]
\[
y_i - (\langle w \cdot x_i \rangle + b) \leq \varepsilon + \hat{\xi}_i.
\]
\[
\xi_i, \hat{\xi}_i \geq 0 \quad \text{for } i = 1, \ldots, n.
\]

The dual can be found by finding the Lagrangian function for the primal problem, differentiating this function with respect to the primal variables, and substituting equivalent expression for the primal variables in the Lagrangian function. The resulting dual formulation is given as,
\[
\max \quad -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_i - \alpha_i) \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \\
- \varepsilon \sum_{i=1}^{n} (\alpha_i + \hat{\alpha}_i) + \sum_{i=1}^{n} y_i (\hat{\alpha}_i - \alpha_i)
\]

subject to \[
\sum_{i=1}^{n} (\alpha_i - \hat{\alpha}_i) = 0 \\
0 \leq \alpha_i, \hat{\alpha}_i \leq C \quad \text{for } i = 1, \ldots, n.
\]

From the solution \(\alpha^*\) and \(\hat{\alpha}^*\), the regression function is as in (2–66), however \(b^*\) is chosen such that \(f(\mathbf{x}_i) - y_i = -\varepsilon\) for any \(i\) with \(0 < (\hat{\alpha}^* - \alpha^*) < C\).
CHAPTER 3
SELECTIVE KERNEL-BASED METHODS

The motivation for the selective kernel-based methods comes from a real life problem, in which time series are to be aligned for improving classification results. The individual pattern vectors and corresponding labels or values from the machine learning context are extended to sets of pattern vectors sharing the same label or value. First, we study the problem of choosing a single pattern vector from each set of possibilities in order to find an optimum separation between the selected positive and negative pattern vectors. This problem, which we call hard selection is highly combinatorial in nature. Therefore, we introduce relaxations of this problem. These soft selection problems, in contrast, slightly differ from the optimization problems we have studied in Chapter 2. However, the general structure of the altered problems is quite similar to the original ones with a convex objective, and linear constraints on a convex domain. This convenience allows us to derive dual formulations in which we can apply the kernel trick to obtain nonlinear classifiers. This chapter includes formulations for the selective versions of support vector machines and support vector regression and their variations. The results on toy problems are presented for the hard selection methods, as well as the soft selection methods, which include incremental elimination and incremental inclusion schemes.

3.1 Selective Support Vector Machines

In the standard support vector machine framework, the input for the classifier is the set of labeled pattern vectors in the training set \( s = \{(x_1, y_1), \ldots, (x_n, y_n)\} \). Now, we consider that instead of each pattern vector, there is a set \( X_i = \{x_{i,1}, \ldots, x_{i,t}\} \) of \( t \) pattern vectors with the same label \( y_i \), where the set \( X_i \) can be considered as a \( t \times d \) matrix, each row being a pattern vector in \( \mathbb{R}^d \). An example is shown in Figure 3-1.

The first type of problem we want to solve is the hard selection problem, in which a single pattern \( x_{i,k^*} \in X_i \) is selected for each set such that the margin for the selected patterns is maximized. We give the formal definition below.
Definition 3.1.1 (Hard Selection Problem). Let \( X = \{X_1, \ldots, X_n\} \) be sets of pattern vectors with \( t \) pattern vectors \( x_{i,1}, \ldots, x_{i,t} \) in each set \( X_i \), and let \( y = \{y_1, \ldots, y_n\} \) be the corresponding labels for each set with each pattern vector in \( X_i \) having the same label \( y_i \). Choose exactly one pattern vector \( x_{i,j}^{*} \) from each set \( X_i \) such that the margin between the selected pattern vectors with label \(+1\) and \(-1\) is maximized.

The hard selection problem can be formulated as a mixed integer quadratic optimization problem as follows, which involves a modification over the standard support vector machine optimization problem.

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \xi_{i,k}^2 \\
\text{subject to} & \quad y_i (\langle w \cdot x_i \rangle + b) \geq 1 - \xi_{i,k} - M(1 - v_{i,k}) \quad i = 1, \ldots, n; \; k = 1, \ldots, t, \\
& \quad \sum_{k=1}^{t} v_{i,k} = 1 \quad i = 1, \ldots, n \\
& \quad v_{i,k} \in \{0, 1\} \quad i = 1, \ldots, n; \; k = 1, \ldots, t.
\end{align*}
\]

In (3–1), in the first set of constraints (3–1), \( M \) is a large positive number. This can be set as the maximum distance between any pair of pattern vectors in \( X \). When the corresponding \( v_{i,k} = 0 \), then, basically, this constraint has no effect on the problem, and
it is equivalent to removing the pattern vector from the training set. When \( \nu_i = 1 \), the constraint reduces to a regular constraint in the standard SVM formulation. The second set of constraints (3–1) ensure that only one pattern vector is chosen from each set of points. Although this problem finds the optimal subset of pattern vectors for the positive and the negative class, it is highly combinatorial.

We relax the binary variables in the hard selection problem in such a way that they would give us a measure on how much slack does a pattern vector require if it is misclassified. The following question is, how can we diminish the effect of misclassified pattern vectors on the objective function in order to have a large margin? We attempt to answer these questions with different relaxations of the hard selection problem, in which binary variables are replaced with continuous variables serving as free slack variables for pattern vectors that are misclassified.

We know from Section 2.4.2 that in the soft margin classification, each pattern vector has a slack variable, but this slack is penalized in the objective function. Here, we introduce the concept of restricted free slack for those pattern vectors which are close to the separating hyperplane, or misclassified. The main motivation behind this relaxation is that such pattern vectors are the ones which impose the orientation and placement of the separating hyperplane. We want to diminish their effect of these pattern vectors and let the pattern vectors which are more separated determine the orientation and placement of the separating hyperplane.

In Figure 3-2, there are pairs of red pattern vectors and blue pattern vectors. If we ignore the pairs and consider the problem as a standard classification problem, since blue and red patterns are separable, a standard support vector machine algorithm would solve this problem without using any slack, however with a small margin. It is intuitive and straightforward that if the pattern vectors towards the middle were to be removed, the remaining pattern vectors would have a large margin. By introducing free slack variables we can diminish the effect of the middle group of pattern vectors, however we restrict the
total slack to avoid trivial solutions. The reason we call them free slack is because they are not penalized in the objective function. We still keep the original slack variables which are penalized for the problem to stay feasible for any training set. we would like to note that, given a hyperplane, if a pattern vector is misclassified, then the free slack will be used first depending on how much free slack is available.

Figure 3-2. Classification of a separable set in the presence of restricted free slack

There are two problems to be addressed. The first one is: how much slack should be provided?, and the second one is: how to distribute the total slack? For the first problem, an intuitive approach can be adapted. In the standard support vector machine formulation, a functional margin of 1 is required. Since we have at least two pattern vectors per set, we prefer to provide a total free slack of $n$, which is a free slack of 1 per set on the average. For a set of two pattern vectors, there are three possibilities: i) both are classified right, ii) one of them is misclassified, iii) both are misclassified. In the first case the pattern vector that is closer to the separating hyperplane, in the second case the misclassified pattern vector, and in the third case the pattern vector further from the hyperplane would require more free slack compared to the other pattern vector.

For distributing the restricted free slack, we introduce two approaches. In the first approach, there is a total slack of $n$, and each pattern vector is allowed to receive slack irrespective of the set they belong to. We refer to this case as pooled free slack (PFS). Figure 3-3 shows the distribution for PFS. In the second approach, however, each set of
pattern vector receive an equal amount of 1 regardless. We refer to the second case as free slack per set (FSS). The distribution of FSS is more restrictive therefore its effect on the separating hyperplane is not as much as the PFS, as seen in Figure 3-4. In the following subsections, we present modified optimization problems together with their dual representations in order to study selective kernel methods in linear and nonlinear classification problems.

![Figure 3-3. Distribution of pooled free slack (PFS)](image)

![Figure 3-4. Distribution of free slack per set (FSS)](image)

### 3.1.1 Selective 2-Norm Support Vector Machines

Assume that there are sets $X_i$, $i = 1, \ldots, n$ of vectors, $x_{i,k}$, $k = 1, \ldots, t$. The following optimization problem finds the maximal margin with PFS approach.
minimize \( \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \xi_{i,k}^2 \) (3–1)

subject to \( y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_{i,k} - \nu_{i,k} \) \( i = 1, \ldots, n; \quad k = 1, \ldots, t \)

\( \sum_{i=1}^{n} \sum_{k=1}^{t} \nu_{i,k} \leq n \) (3–2)

\( \nu_{i,k} \geq 0 \) \( i = 1, \ldots, n; \quad k = 1, \ldots, t \).

The Lagrangian function for (3–1) is given as follows,

\[
L(\mathbf{w}, b, \xi, \tilde{\xi}) = \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \xi_{i,k}^2 \\
- \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k} [y_i(\langle \mathbf{w} \cdot \mathbf{x}_{i,k} \rangle + b) - 1 + \xi_{i,k}] \\
- \beta (n - \sum_{i=1}^{n} \sum_{k=1}^{t} \nu_{i,k}) \\
- \sum_{i=1}^{n} \sum_{k=1}^{t} r_{i,k} \nu_{i,k}.
\] (3–3)

In this equation, \( \alpha, \beta \) and \( r \) are Lagrangian multipliers. Taking partial derivatives with respect to the primal variables \( \mathbf{w}, b, \) and each \( \xi_i \) and \( \nu_i \), and imposing stationarity, we get the following expressions:

\[
\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i = 0
\] (3–5)

\[
\frac{\partial L}{\partial b} = \sum_{i=1}^{n} y_i a_i = 0
\] (3–6)

\[
\frac{\partial L}{\partial \xi_{i,k}} = C \xi_{i,k} - \alpha_i = 0
\] (3–7)

\[
\frac{\partial L}{\partial \nu_{i,k}} = \beta - \alpha_{i,k} - r_{i,k}.
\] (3–8)

Substituting the partial derivatives (3–5 - 3–8) in (3–3), we can write the following dual formulation:
\[
\text{max } \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} y_i y_j \alpha_{i,k} \alpha_{j,l} \langle x_{i,k}, x_{j,l} \rangle - \frac{1}{2C} \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k}^2 - \beta n
\]

subject to
\[
\sum_{i=1}^{n} \sum_{k=1}^{t} y_i \alpha_{i,k} = 0
\]
\[
0 \leq \alpha_{i,k} \leq \beta \quad \text{for } i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]

Note that kernel induced maps to nonlinear feature spaces can be used by replacing
the linear dot product \(\langle x_{i,k}, x_{j,l} \rangle\) in (3–9) with a kernel \(K(x_{i,k}, x_{j,l})\).

For the FSS case, the constraint (3–2) becomes,
\[
\sum_{k=1}^{t} u_{i,k} \leq 1 \quad i = 1, \ldots, n.
\]

Consequently, this change introduces a separate Lagrangian multiplier \(\beta_i\) for each
constraint, and therefore the relevant component (3–4) in the Lagrangian function changes
to,
\[
- \sum_{i=1}^{n} \beta_i \left(1 - \sum_{k=1}^{t} u_{i,k}\right)
\]

Following the differentiation of the Lagrange function, imposing stationarity and
substituting the expressions back in the Lagrangian, we can obtain the dual formulation
for the free slack per set case as,
\[
\max \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} y_i y_j \alpha_{i,k} \alpha_{j,l} \langle x_{i,k}, x_{j,l} \rangle - \frac{1}{2C} \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k}^2 - \sum_{i=1}^{n} \beta_i \\
\text{subject to} \sum_{i=1}^{n} \sum_{k=1}^{t} y_i \alpha_{i,k} = 0 \\
0 \leq \alpha_{i,k} \leq \beta_i \quad \text{for } i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]

3.1.2 Selective 1-norm Support Vector Machines

An alternative formulation for support vector machines is to penalize 1-norm penalty term, as shown in (2–39). Here we present the selective version of the 1-norm SVM formulations. The derivation of the primal and dual optimization problem are similar to those discussed for the 2-norm case, therefore we briefly present these formulations. The primal for the PFS approach is as follows.

\[
\text{minimize} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \sum_{k=1}^{t} \xi_{i,k} \\
\text{subject to} \sum_{i=1}^{n} \sum_{k=1}^{t} y_i (\langle w \cdot x_i \rangle + b) \geq 1 - \xi_{i,k} - \nu_{i,k} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t \\
\sum_{i=1}^{n} \sum_{k=1}^{t} \nu_{i,k} \leq n \\
\xi_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \\
\nu_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]

As in the 2-norm case, we derive the Lagrangian function for the primal problem, differentiate it with respect to the primal variables, impose stationarity and substitute
the resulting expressions back in the Lagrangian function to obtain the following dual formulation.

\[
\text{maximize } \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} y_i y_j \alpha_{i,k} \alpha_{j,l} \langle x_{i,k} \cdot x_{j,l} \rangle - \beta n \tag{3-15}
\]

subject to

\[
\sum_{i=1}^{n} \sum_{k=1}^{t} y_i \alpha_{i,k} = 0
\]

\[
0 \leq \alpha_{i,k} \leq \beta \leq C \quad \text{for } i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]

In (3–15), the dot product \( \langle x_i \cdot x_j \rangle \) can be replaced with a kernel \( K(x_i, x_j) \) for a nonlinear map to the kernel induced feature space.

The primal and dual formulations for the 1-norm FSS case are slightly different from the general free slack case, where the general free slack of \( n \) in constraint (3–14), now implies that the free slack is restricted to 1 for each set, which is given as,

\[
\text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \sum_{k=1}^{t} \xi_{i,k} \tag{3–16}
\]

subject to

\[
y_i(\langle w \cdot x_i \rangle + b) \geq 1 - \xi_{i,k} - v_{i,k} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t
\]

\[
\sum_{k=1}^{t} v_{i,k} \leq 1 \quad i = 1, \ldots, n; \tag{3–17}
\]

\[
\xi_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]

\[
v_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]

Through standard Lagrangian function, differentiation and substitution method, we can obtain the following corresponding dual problem:
maximize \[ \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} y_i y_j \alpha_{i,k} \alpha_{j,l} \langle x_{i,k} \cdot x_{j,l} \rangle - \sum_{i=1}^{n} \beta_i \] (3–18)

subject to
\[ \sum_{i=1}^{n} \sum_{k=1}^{t} y_i \alpha_{i,k} = 0 \]

\[ 0 \leq \alpha_{i,k} \leq \beta_i \leq C \quad \text{for} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \]

Having introduced 2-norm and 1-norm formulations of selective support vector machines with the PFS and FSS case, we can move on how we use these optimization problems.

### 3.1.3 Selecting Pattern Vectors with Large Margin

Our objective may differ with respect to minimum or maximum values of the free slack assigned to pattern vectors. Based on such values we can adopt two immediate approaches for finding the desired points. The first one is to choose the pattern vector with the lowest free slack from each seat. This requires the optimization problem to be solved only once. We refer to this approach as *choose best vectors*. The alternative approach is to identify those pattern vectors that are very close to the separating hyperplane, or those which are misclassified, and possibly incurring a high penalty. From the solution to the optimization problem we remove the pattern vectors with the highest free slack from each set, and resolve the problem with the reduced set. We refer to this approach as *eliminate worst vectors*. This approach requires the optimization problem to be solved \( t - 1 \) times, until there is one pattern vector remaining from each set.

Through empirical tests on toy examples, eliminate worst method outperformed choose best method. Eliminate worst method approaches to a linear separation for separable sets. For an appropriate parameter value, for example \( \sigma \) in the Gaussian kernel, the separating curve turns into a linear hyperplane for sufficiently large \( \sigma \) value. In Figure 3-5 there are three pattern vectors in each set and the groups themselves are separate.
In both standard SVM and selective SVM, lower values of Gaussian bandwidth causes overfitting. However, it is clear that for the higher values of Gaussian kernel bandwidth that standard SVM creates the hyperplane that separates all red and all blue pattern vectors from each other as shown in 3-5. Selective SVM, on the other hand creates a separating hyperplane which align itself such that the margin is maximized with respect to the patterns chosen at the end of the elimination process. Note that these pattern vectors are most separated from each other in Figure 3-6.

Figure 3-5. Standard SVM on a separable 3D example for different kernel bandwidths

### 3.2 Selective Support Vector Regression

The similarity between the SVM classification and SVM regression, which is studied in detail in Chapter 2 suggest a natural extension of selective SVM methods to SVM regression methods. The structure of the regression problems are very similar to the classification problems covered in Section 3.1.

In standard support vector regression, the input is a set of pairs of a pattern vector and a dependent value, \( S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \). In the selective support vector
Figure 3-6. Selective SVM on a separable 3D example: bold points are selected

regression, we have sets $X_i = \{x_{i,1}, \ldots, x_{i,k}\}$ of pattern vectors with corresponding
dependent values for each set $S = \{(X_1, y_1), \ldots, (X_n, y_n)\}$. The main objective is selecting
a single pattern vector from each set such that the objective function of the optimization
problem (2-57) is minimized. As in the SelSVM, we start by defining hard selection for
support vector regression.

**Definition 3.2.1 (SelSVR Hard Selection).** Let $X = \{X_1, \ldots, X_n\}$ be sets of pattern
vectors with $t$ pattern vectors $x_{i,1}, \ldots, x_{i,t}$ in each set $X_i$, and let $y = \{y_1, \ldots, y_n\}$ be the
corresponding values for each set with each pattern vector in $X_i$ having the same value
$y_i$. Choose exactly one pattern vector from each set such that the sum of the $\varepsilon$-insensitive
errors between the pattern vectors and the regression function is minimized.

The hard selection support vector regression problem can be formulated as a
quadratic mixed integer programming problem as follows,
\[
\text{minimize } \frac{1}{2} \| \mathbf{w} \|^2 + C \frac{1}{2} \sum_{i=1}^{n} (\xi_{i,k}^2 + \hat{\xi}_{i,k}^2) \quad (3-19)
\]

subject to

\[
\begin{align*}
(\langle \mathbf{w} \cdot \mathbf{x}_{i,k} \rangle + b) - y_i & \leq \varepsilon + \xi_{i,k} + M(1 - \upsilon_{i,k}) \quad i = 1, \ldots, n; \quad k = 1, \ldots, t \quad (3-20) \\
y_i - (\langle \mathbf{w} \cdot \mathbf{x}_{i,k} \rangle + b) & \leq \varepsilon + \hat{\xi}_{i,k} + M(1 - \hat{\upsilon}_{i,k}) \quad i = 1, \ldots, n; \quad k = 1, \ldots, t \quad (3-21) \\
\sum_{k=1}^{t} \upsilon_{i,k} + \hat{\upsilon}_{i,k} & = 1 \quad (3-22) \\
v_{i,k} & \in \{0, 1\} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \quad (3-23)
\end{align*}
\]

In the above formulation, \( M \) is a large positive number, such that for those points with \( \upsilon_{i,k} = 0 \), the related constraint is always satisfied, and thus, does not have any influence on the problem. This is equivalent to removing this pattern vector from the problem. \( M \) can conveniently be set to the maximum distance between the distances of any two pairs of pattern vectors. Constraints (3–20) and (3–21) account for the case if a pattern vector is above or below the regression function, respectively. Finally, constraint (3–22) ensures that only one of pattern vector from each set is selected. The hard selection problem, like in SelSVR case, is a highly combinatorial problem and there is no efficient solution known.

We consider relaxations of the hard selection problem in order to obtain an efficient and effective solution. Instead of the binary variables, we introduce continuous variables for each pattern vector, total value of which is bounded. This is the same framework of relaxations that we discuss in Section 3.1. Likewise, we include two approaches for the free slack values. In the \textit{pooled free slack} (PFS) case, each pattern vector can be assigned free slack regardless of the other pattern vectors in their set, where as \textit{free slack per set} (FSS) bounds the free slack values for each set with 1.
We cover two formulations based on the penalization of the slack variable, namely selective \(i\) quadratic, and \(ii\) linear \(\varepsilon\)-insensitive loss regression. For each formulation, we consider both the general and free slack per set approaches.

### 3.2.1 Selective 2-norm \(\varepsilon\)-Insensitive Regression

Assume that there are sets \(X_i, i = 1, \ldots, n\) of vectors, \(x_{i,k}, k = 1, \ldots, t\). Then, the following optimization problem finds the regression function that minimizes the loss function, given that the total general free slack for all pattern vector is \(n\varepsilon\).

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + \frac{C}{2} \left( \sum_{i=1}^{n} \sum_{k=1}^{t} (\xi_{i,k}^2 + \hat{\xi}_{i,k}^2) \right) \\
\text{subject to} & \quad (\langle w \cdot x_{i,k} \rangle + b) - y_i \leq \varepsilon + \xi_{i,k} + v_{i,k} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \\
& \quad y_i - (\langle w \cdot x_{i,k} \rangle + b) \leq \varepsilon + \hat{\xi}_{i,k} + \hat{v}_{i,k} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \\
& \quad \sum_{i=1}^{n} \sum_{k=1}^{t} (v_{i,k} + \hat{v}_{i,k}) \leq n\varepsilon \\
& \quad v_{i,k}, \hat{v}_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\end{align*}
\]

The Lagrangian function for (3–24) is given as,

\[
L(w, b, \xi, \hat{\xi}) = \frac{1}{2} \|w\|^2 + \frac{C}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} (\xi_{i,k}^2 + \hat{\xi}_{i,k}^2) - \sum_{i=1}^{n} \sum_{k=1}^{t} \alpha_{i,k} [\varepsilon + \xi_{i,k} + v_{i,k} + y_{i,k} - \langle w \cdot x_{i,k} \rangle - b] \\
- \sum_{i=1}^{n} \sum_{k=1}^{t} \hat{\alpha}_{i,k} [\varepsilon + \hat{\xi}_{i,k} + \hat{v}_{i,k} - y_{i,k} + \langle w \cdot x_{i,k} \rangle + b] \\
- \beta \left[ n\varepsilon - \sum_{i=1}^{n} \sum_{k=1}^{t} (v_{i,k} + \hat{v}_{i,k}) \right] \\
- \sum_{i=1}^{n} \sum_{k=1}^{t} (\eta_i v_{i,k} + \hat{\eta}_i \hat{v}_{i,k})
\]
Differentiating this Lagrangian function with respect to the decision variables from (3–24) $w$, $b$, and each $\xi_i$, $\hat{\xi}_i$, $\upsilon_i$, $\hat{\upsilon}_i$, and imposing stationarity, we get the following expressions:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \sum_{k=1}^{t} (\hat{\alpha}_{i,k} - \alpha_{i,k}) x_{i,k} = 0 \quad (3–27)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} (\alpha_{i,k} - \hat{\alpha}_{i,k}) = 0 \quad (3–28)$$

$$\frac{\partial L}{\partial \xi_{i,k}} = C \xi_{i,k} - \alpha_{i,k} = 0 \quad (3–29)$$

$$\frac{\partial L}{\partial \hat{\xi}_{i,k}} = C \hat{\xi}_{i,k} - \hat{\alpha}_{i,k} = 0 \quad (3–30)$$

$$\frac{\partial L}{\partial \upsilon_{i,k}} = \beta - \alpha_{i,k} - \eta_i = 0 \quad (3–31)$$

$$\frac{\partial L}{\partial \hat{\upsilon}_{i,k}} = \beta - \hat{\alpha}_{i,k} - \hat{\eta}_i = 0 \quad (3–32)$$

Substituting the expressions (3–27 - 3–32) in (3–26), we obtain the following dual formulation:

$$\text{maximize} \quad -\frac{1}{2C} \sum_{i=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k}^2 + \hat{\alpha}_{i,k}^2)$$

$$\quad - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} (\hat{\alpha}_{i,k} - \alpha_{i,k})(\hat{\alpha}_{j,l} - \alpha_{j,l})(x_{i,k} \cdot x_{j,l})$$

$$\quad - \varepsilon \sum_{i=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} + \hat{\alpha}_{i,k}) + \sum_{i=1}^{n} \sum_{k=1}^{t} y_{i,k}(\hat{\alpha}_{i,k} - \alpha_{i,k}) - \beta n \varepsilon$$

subject to

$$\sum_{j=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} - \hat{\alpha}_{i,k}) = 0$$

$$0 \leq \alpha_{i,k} \leq \beta \quad \text{for } i = 1, \ldots, n; k = 1, \ldots, t$$

$$0 \leq \hat{\alpha}_{i,k} \leq \beta \quad \text{for } i = 1, \ldots, n; k = 1, \ldots, t.$$
From the standard kernel trick, the dot product $\langle x_i, x_j \rangle$ can be replaced with a kernel $K(x_i, x_j)$ for nonlinear regression.

The free slack per set approach results in a similar formulation, where constraint (3–39) is replaced by the following constraint for each set $i$, which limits the free slack to $\varepsilon$ per set.

$$\sum_{k=1}^{t} v_{i,k} + \hat{v}_{i,k} \leq \varepsilon \quad i = 1, \ldots, n. \quad (3–34)$$

This change reflects on the Lagrangian function such that the term

$$- \beta \left[ n \varepsilon - \sum_{i=1}^{n} \sum_{k=1}^{t} (v_{i,k} + \hat{v}_{i,k}) \right] \quad (3–35)$$

is replaced with,

$$- \varepsilon \sum_{i=1}^{n} \beta_i + \sum_{i=1}^{n} \sum_{k=1}^{t} \beta_i (v_{i,k} + \hat{v}_{i,k}). \quad (3–36)$$

The resulting dual formulation for the free slack per set is given as,

$$\text{maximize} \quad - \frac{1}{2C} \sum_{i=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k}^2 + \hat{\alpha}_{i,k}^2)$$

$$- \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} (\hat{\alpha}_{i,k} - \alpha_{i,k})(\hat{\alpha}_{j,l} - \alpha_{j,l}) \langle x_{i,k}, x_{j,l} \rangle$$

$$- \varepsilon \sum_{i=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} + \hat{\alpha}_{i,k}) + \sum_{i=1}^{n} \sum_{k=1}^{t} y_i (\hat{\alpha}_{i,k} - \alpha_{i,k}) - \varepsilon \sum_{i=1}^{n} \beta_i$$

subject to

$$\sum_{j=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} - \hat{\alpha}_{i,k}) = 0$$

$$0 \leq \alpha_{i,k} \leq \beta_i \quad \text{for } i = 1, \ldots, n; k = 1, \ldots, t$$

$$0 \leq \hat{\alpha}_{i,k} \leq \beta_i \quad \text{for } i = 1, \ldots, n; k = 1, \ldots, t.$$
The dot product $\langle x_i, x_j \rangle$ can be replaced with a kernel $K(x_i, x_j)$ for nonlinear regression.

### 3.2.2 Selective 1-norm $\epsilon$-Insensitive Regression

The second alternative as in the SelSVR problem is to penalize the 1-norm of the non-free slack for each pattern vector. Because of the similarity to the 2-norm case, we present only the primal and dual formulations, and skip the intermediate steps. The primal problem for selective 1-norm $\epsilon$-insensitive regression is given as,

$$
\begin{align*}
& \text{minimize} \quad \frac{1}{2} \| w \|^2 + \frac{C}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} (\xi_{i,k} + \hat{\xi}_{i,k}) \\
& \text{subject to} \quad \begin{aligned}
& (\langle w \cdot x_{i,k} + b \rangle - y_i) \leq \varepsilon + \xi_{i,k} + v_{i,k} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \\
& y_i - (\langle w \cdot x_{i,k} + b \rangle) \leq \varepsilon + \hat{\xi}_{i,k} + \hat{v}_{i,k}. \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \\
& \sum_{i=1}^{n} \sum_{k=1}^{t} (v_{i,k} + \hat{v}_{i,k}) \leq n \varepsilon \\
& \xi_{i,k}, \hat{\xi}_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t. \\
& v_{i,k}, \hat{v}_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\end{aligned}
\end{align*}
$$

(3-38)

(3-39)

Deriving the Lagrangian function, differentiating it with respect to the decision variables in the primal problem, imposing stationarity and substituting the resulting expressions back in the Lagrangian function we obtain the following dual problem.
maximize \[-\frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} (\hat{\alpha}_{i,k} - \alpha_{i,k})(\hat{\alpha}_{j,l} - \alpha_{j,l})(\mathbf{x}_{i,k} \cdot \mathbf{x}_{j,l}) \]
\[-\varepsilon \sum_{i=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} + \hat{\alpha}_{i,k}) + \sum_{i=1}^{n} \sum_{k=1}^{t} y_i(\hat{\alpha}_{i,k} - \alpha_{i,k}) - \beta n\varepsilon \]
subject to
\[
\sum_{j=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} - \hat{\alpha}_{i,k}) = 0
\]
\[0 \leq \alpha_{i,k} \leq \beta \leq C \text{ for } i = 1, \ldots, n; k = 1, \ldots, t \]
\[0 \leq \hat{\alpha}_{i,k} \leq \beta \leq C \text{ for } i = 1, \ldots, n; k = 1, \ldots, t.\]

In the free slack per set approach, we allow a free slack of \(\varepsilon\) for each set. This means that the constraint regarding the free slack needs to be separate for each set of points. The primal problem for this variant is given as,

\[
\text{minimize } \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} (\xi_{i,k} + \hat{\xi}_{i,k}) \quad (3-40)
\]
subject to
\[
(\langle \mathbf{w} \cdot \mathbf{x}_{i,k} \rangle + b) - y_i \leq \varepsilon + \xi_{i,k} + v_{i,k} \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]
\[y_i - (\langle \mathbf{w} \cdot \mathbf{x}_{i,k} \rangle + b) \leq \varepsilon + \hat{\xi}_{i,k} + \hat{v}_{i,k}. \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]
\[
\sum_{k=1}^{t} (v_{i,k} + \hat{v}_{i,k}) \leq \varepsilon \quad i = 1, \ldots, n\varepsilon
\]
\[
\xi_{i,k}, \hat{\xi}_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]
\[v_{i,k}, \hat{v}_{i,k} \geq 0 \quad i = 1, \ldots, n; \quad k = 1, \ldots, t.
\]

Through standard Lagrangian function, differentiation, optimality conditions and substitution, we obtain the dual formulation given as,
maximize \(-\frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{t} \sum_{j=1}^{n} \sum_{l=1}^{t} (\hat{\alpha}_{i,k} - \alpha_{i,k})(\hat{\alpha}_{j,l} - \alpha_{j,l}) \langle x_{i,k} \cdot x_{j,l} \rangle \)

\[-\varepsilon \sum_{i=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} + \hat{\alpha}_{i,k}) + \sum_{i=1}^{n} \sum_{k=1}^{t} y_i(\hat{\alpha}_{i,k} - \alpha_{i,k}) - \varepsilon \sum_{i=1}^{n} \beta_i \]

subject to

\[\sum_{j=1}^{n} \sum_{k=1}^{t} (\alpha_{i,k} - \hat{\alpha}_{i,k}) = 0\]

\[0 \leq \alpha_{i,k} \leq \beta_i \leq C \quad \text{for} \quad i = 1, \ldots, n; k = 1, \ldots, t\]

\[0 \leq \hat{\alpha}_{i,k} \leq \beta_i \leq C \quad \text{for} \quad i = 1, \ldots, n; k = 1, \ldots, t.\]

In both of the dual formulations for the 1-norm case, the dot product \(\langle x_i \cdot x_j \rangle\) can be replaced by the kernel \(K(x_i, x_j)\) for nonlinear regression.

We tested the efficiency of selective SVM regression on a 3D surface which is given by points \((x, y, z)\) such that \(z = \sin(r)/r\), where \(r = \sqrt{x^2 + y^2}\). The selective SVR is compared with a naïve method in which regular SVR is used and whichever point was closer to the predicted surface, was included in the winning set. For selective SVR winning points are those chosen by the elimination method. The two classes of points come from the underlying function plus noise. In the first group of pattern vectors, \(x\) and \(y\) coordinates were distorted with a uniform noise within a box of 1 by 1 unit, whereas the second set of pattern vectors’ coordinates were distorted uniformly within a box of 3 by 3 units. The results are encouraging in the sense that although there were inconclusive cases between which point to be picked, after settling a threshold value on the amount of slack required by a pair of points, the ratio of blue point (with less noise) selected to red points selected always dominated the naïve method, as given in Figure 3-7.
Figure 3-7. Selective SVR compared to a naïve method for different bandwidths
In the previous chapters we reviewed the kernel-based machine learning algorithms, more specifically support vector machine (SVM) methods, which include classification, feature selection and regression methods. We introduced novel selective classification and regression methods. This chapter involves the neural data set on which the methods we have developed so far are applied. In a fact, this data set is the main motivation to apply kernel-based machine learning algorithms on neural data.

The data set consist of local field potentials (LFP) from primate brains during a visual discrimination task. We give further detail on how these experiments were performed, including the technical details and the experimental paradigm. Then we introduce the SVM methods for classification and feature selection. The results from the application of these methods are reported and illustrated for three macaque monkeys. The selective classification methods are introduced to account for the temporal variations among the many repetitions of the same cognitive task which are referred to as single trials. The improved results after the application of selective SVM methods are also presented.

4.1 Visual Discrimination Task

The experiments to acquire the data used in this study takes place in the Laboratory of Neuropsychology at the NIMH between 1984-1988. The data are collected from three young adult macaque monkeys (Macaca mulatta) GE, LU and TI. This data has previously been used in a number of studies [57–60]. All three monkeys are well trained to respond to a go/nogo visual stimuli before the electrodes are placed in these monkeys’ brains. Surgeries to place the electrodes are performed under sterile conditions with the monkey under general anesthesia. Teflon coated platinum-iridium wire bipolar electrodes with a 0.125 mm diameter are inserted in the hemisphere contralateral to the hand used in the task, with the less advanced tip of the electrodes extended 0.5 mm into the dura
and the more advanced tip extended 2.5 mm into the cortex. For more details, see [60]. In Figure 4-1 the electrode locations that are used in this study are shown.

![Electrode Locations](image)

Figure 4-1. Approximate placement of electrodes in the tree monkeys

Local field potential (LFP) data are collected from each monkey during a number of sessions. Each session consists around 1000 trials. The data are band-pass filtered between 1 and 100 Hz and digitized at 200 Hz. For each trial, data are recorded for 900 ms after the monkey initiates the trial. The time between the trials are around a second.

The stimuli are designed to create lines and diamonds using eight squares as shown in Figure 4-2. The stimuli are referred as right slanted line and right slanted diamond in the first row, and left slanted line and left slanted diamond in the second row. Note that lines are given in the first column and diamonds are given in the second column. The go stimuli is chosen to be either both lines or both diamonds, and never any other possible combination of the four stimulus types. Note that when the two line stimuli and the two diamond stimuli are superimposed onto each other separately, the resulting two shapes are identical. Therefore, the monkey has to distinguish a line from a diamond by correctly realizing the placement of at least two squares. The total area, contrast, edge length and brightness are constant across the stimuli types and trials. The stimuli is placed 57 cm from the subject and its overall size is 6 degrees between the outer corner squares. The presentation time is computer controlled with a piezoelectric shutter.

A trial is initiated by the monkey pressing a lever and keeping it pressed. The initiation is followed by a random amount of time, uniformly distributed between 1200
and 2200 ms, before the stimulus appears on the screen for exactly 100 ms. The monkey is expected to respond to a stimulus within the 500 ms following the stimulus onset. The release of the lever is the correct go response, and keeping it pressed is the correct nogo response. The correct go responses are rewarded with a small amount of water, whereas correct nogo responses are not rewarded. The outline of the task is presented in Figure 4-3.

In all sessions, there are two different stimulus-response contingencies: line-go/diamond-nogo and diamond-go/line-nogo. In this study, only correct go and correct nogo trials are considered. For each trial LFP data recording starts 90 ms prior to the stimulus onset.
and continues for 600 ms. After the digitization at 200 Hz, each electrode produces 120 discrete recordings with 5 ms between two consecutive recordings, for each trial.

4.2 Methods

Support Vector Machines (SVMs) are the state-of-the-art methods in machine learning literature. Although it is a relatively new methodology that emerged within the last decade, its application spectrum covers a rich variety of problems. The most general application of SVMs is classification. There are recent studies on the classification of neural data using SVMs [8, 9]. Considering the multi-dimensional nature of the problem and the nonlinear behavior of the brain, SVM classification could be used on each time point to detect different stages of early effects of visual stimulus, discrimination of the stimuli type, and preparation and execution of the response. Moreover, the contribution of the channels to the classification at each time point would reveal valuable information on the spatiotemporal interaction between the visual system and the motor system. In the following subsections the SVM method is briefly introduced, followed by the SVM based feature selection technique applied in this study.

4.3 Classification and Channel Selection Results

In this work we are interested in detecting the spatiotemporal effects in the visual and motor cortexes of a macaque monkey’s brain during a visual discrimination task. To obtain these results, we have distinguished different sets of labels, considering a different effect for each label. In the following table different labels are associated with different effects that are to be detected. We distinguish between three effects: i) early effect of visual stimulus, ii) discrimination of stimulus, and iii) response preparation and execution. For each effect type, binary SVM classification has been applied by labeling the trials in different combinations.

The reason in associating these labels to the different effects are as follows: 1) For the early visual effect, the detection of the direction of the stimuli (left or right). 2) For the discrimination of the visual stimuli, the monkey has to distinguish between a line and a
Table 4-1. Different stages of the visuomotor task

<table>
<thead>
<tr>
<th>Stage</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early effect of visual stimulus</td>
<td>Left vs. Right stimuli</td>
</tr>
<tr>
<td>Discrimination of stimulus</td>
<td>Line vs. Diamond stimuli</td>
</tr>
<tr>
<td>Response preparation and execution</td>
<td>Go vs. Nogo</td>
</tr>
</tbody>
</table>

diamond stimuli. In the motion part the main effect is the preparation of the impulse sent from monkey’s brain to its hand and its execution.

We are consider two sets of data, where in the first one lines are go stimuli and the diamonds are nogo stimuli. In the second one diamonds are go stimuli and lines are nogo stimuli. For each monkey we have around 10,000 trials. Since the combinatorial result of the classification scheme, and the number of time points is computationally demanding, for each monkey, between 20% and 30% of the available trials were used to perform the classification. The trials are randomly selected from a combined set of data involving all the trials from all the session. We group the trials in various ways with respect to their stimulus type. If we let a trial with right line, right diamond, left line and left diamond be marked as $s_1^r, s_2^r, s_3^r$ and $s_4^r$, respectively, coming from data set $i = 1, 2$, then we can form three different sets of data with respect to our interest. The possible combinations are:

1. RIGHT vs LEFT $\equiv (S^+ = s_1^r \cup s_2^r \cup s_3^r \cup s_2^l \cup s_3^l \cup s_4^r \cup s_4^l$ vs. $S^- = s_1^l \cup s_2^l \cup s_3^l \cup s_4^l)$.
2. LINE vs DIAMOND $\equiv (S^+ = s_1^l \cup s_2^l \cup s_2^r \cup s_3^l \cup s_4^r \cup s_4^l$ vs. $S^- = s_1^r \cup s_3^r \cup s_3^l \cup s_4^r \cup s_4^l)$,
3. GO vs NOGO $\equiv (S^+ = s_1^1 \cup s_3^1 \cup s_1^2 \cup s_3^2 \cup s_2^1 \cup s_2^2$ vs. $S^- = s_2^1 \cup s_1^1 \cup s_1^2 \cup s_2^2$).

In Figures 4-4, 4-5, and 4-6, the classification and the feature selection results are given for the monkeys GE, LU and TI. The three columns of graphs in each figure correspond to the three different stages we are interested as listed in Table 4.3. In the first row of graphs in each figure, the overall accuracy results are presented. In the second row, the contribution of each channel to the overall classification are given. we refer to these plots as channel selection plots. The channel names and corresponding cortical regions are given in Table F. Knowing the contribution of each channel for a time point $t$, we can sort the channels in descending order of their contribution. In the plots in the third
Table 4-2. Channel names and locations for TI, LU and GE

<table>
<thead>
<tr>
<th>N chn</th>
<th>TI</th>
<th>LU</th>
<th>GE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>StrB Striate</td>
<td>Supr</td>
<td>StrA Striate A</td>
</tr>
<tr>
<td>2</td>
<td>PstA Prestriate</td>
<td>StrF Striate</td>
<td>StrB Striate B</td>
</tr>
<tr>
<td>3</td>
<td>PstB Prestriate</td>
<td>PstS Prestriate</td>
<td>StrC Striate C</td>
</tr>
<tr>
<td>4</td>
<td>AudA</td>
<td>ParE Parietal</td>
<td>PstA Prestriate A</td>
</tr>
<tr>
<td>5</td>
<td>IftB Inferotemporal</td>
<td>AudB</td>
<td>PstB Prestriate B</td>
</tr>
<tr>
<td>6</td>
<td>MotH Motor</td>
<td>FrnD Frontal</td>
<td>PstC Prestriate C</td>
</tr>
<tr>
<td>7</td>
<td>FrnB Frontal</td>
<td>SomV Somatosensory</td>
<td>FefA FEF</td>
</tr>
<tr>
<td>8</td>
<td>StrF Striate</td>
<td>SomM Somatosensory</td>
<td>MotB Motor</td>
</tr>
<tr>
<td>9</td>
<td>IftA Inferotemporal</td>
<td>MotH Motor</td>
<td>SomA Somatosensory</td>
</tr>
<tr>
<td>10</td>
<td>SupT</td>
<td>StrB Striate</td>
<td>ParD Parietal</td>
</tr>
<tr>
<td>11</td>
<td>FrnD Frontal</td>
<td>StrC Striate</td>
<td>ParE Parietal</td>
</tr>
<tr>
<td>12</td>
<td>FefA FEF</td>
<td>IftA Inferotemporal</td>
<td>PrmA Premotor</td>
</tr>
<tr>
<td>13</td>
<td>InPr Inferior-Parietal</td>
<td>AudA</td>
<td>PrmC Premotor</td>
</tr>
<tr>
<td>14</td>
<td>Supr</td>
<td>FrnA Frontal</td>
<td>FrnB Frontal</td>
</tr>
<tr>
<td>15</td>
<td>/</td>
<td>/</td>
<td>FrnG Frontal</td>
</tr>
</tbody>
</table>

row, for each time point we show the change in classification accuracy as we incrementally introduce the sorted channels. The horizontal axis is the time, and the vertical axis shows the number of channels included. We refer to this plot as the incremental accuracy plot.

Results for GE.

RIGHT vs LEFT: The early visual effect, i.e. the right-left separation, is experienced around 110 ms after the stimulus onset with a very sharp peak. The separation between trials labeled as right and left diminishes slowly, reaching a dip at 240 ms, and makes another peak around 300 ms, which diminishes again back to an undetectable level. This behavior can clearly be observed from the feature selection plot, in which channel 5 is clearly very significant wherever there are peaks in the accuracy plot in A. Also, the plot, in which the sorted features are added one by one to the classification looks almost not effected starting with only one feature. This means that only one channel would be enough to do the classification in this case, which supports the results of the feature selection plot.
LINE vs DIAMOND: The line-diamond case is almost in parallel with the right-left case except for a small shift of the dip between the two peaks, and the second peak occurring around 210 ms and 280 ms, respectively. However, the feature selection for this case is not dependent on a single channel. The spots that have a high contrast for a short time interval are from channel 11 and 3 corresponding to the first peak, and channels 5 and 6 corresponding to the second peak. This observation is also supported by the increasing number of features included in the classification.

GO vs NOGO: Although the go-nogo case does not have as sharp peaks as in the previous two cases, there is a sharp increase with a first peak at 200 ms, followed by two other dips at 250 ms and 360 ms, and with slight dips in between the peaks (A, go/nogo). After 360 ms it diminishes and continues as a plateau at a significantly high classification percentage rate. The significant channels that contribute to the classification are 9 and 15 almost throughout the time line, and 5 and 6 for the first peak, 9, 10 and 12 for the second peak, and 9, 10 and 15 for the last peak (B, go/nogo). Including very few number of best channels performs well in classification, however, as more channels are included the accuracy drops slightly, possibly because of the increased noise. As more channels are added, a combination of channels help classification and increase the accuracy again.

Results for LU.

RIGHT vs LEFT: The early visual effect can be seen starting at 100 ms. with a sharp peak at 110 ms and drops back to a low level around 200 ms. Channel 11 contributes to the classification the most and it is the only significant channel at the feature selection plot. This is also verified by the incremental best features plot the figure such that the addition of new features does not change the classification accuracy.

LINE vs DIAMOND: There is no peak at around 100 ms but the accuracy starts to increase, reaching its first peak at around 160 ms. There are two other peaks with higher accuracy at 270 ms and 420 ms. Channel 8 is the most significant channel in the feature selection plot with high intensities around the peaks in the overall accuracy plot. From the
bottom plot, it is observed that increasing the number of best features introduces noise and decrease the classification accuracy.

**GO vs NOGO:** In this case, there is a classification rate around 68 percent between the go and nogo cases even prior to the visual stimulus onset. The first increase in the accuracy appears at around 280 ms, followed by a dip at 240 ms, and a high peak at 350 ms. The most significant channel is 8 in this case. Channel 2 and 14 creates noise and decrease the accuracy between the interval that corresponds to the dip in the overall accuracy graph. This effect is also verified by the incremental features graph; the accuracy is the highest with the best channels, and it starts dropping as more channels are included, however starts increasing as even more channels are introduced.

**Results for TI.**

**RIGHT vs LEFT:** The only sharp increase is at around 110 ms followed by a steady decrease. Channel 2 is the only significant channel in the feature selection plot. Therefore, adding more channels does not change the classification accuracy as seen in the incremental best features graph.

**LINE vs DIAMOND:** The accuracy increases slightly at 100 ms and stays the same until 240 ms, at which point there is another slight increase, reaching its highest value, until 280 ms. The only significant channel is 6, whose highest interval corresponds to the highest point in the overall accuracy graph. The significance of the other channels over any time interval are indistinguishable from the feature selection graph.

**GO vs NOGO:** In this case, there are two highly significant peaks. The first one starts at 150 ms and reaches its maximum at 200 ms, followed by a dip with the lowest point at 260 ms. The second peak starts at 300 ms and reaches its highest point at 350 ms. Channel 8 is the most significant channel, except for the interval where the dip occurs. The classification accuracy decreases as more channels are included because of the noise introduced, however further increasing the number of channels increases the accuracy slightly.
In Figure 4-7 comparative snapshots of peak activities are given for three monkeys together with the times. The location and the time are consistent for all three stages.

4.4 Time Series Alignment with Dynamic Time Warping

Brain is a very complex and highly dynamic system. Although the result of the go-nogo experimentation we study is identical in the sense that the desired outcome is met, the progression of stages involved in this visuomotor task may vary over time. This phenomenon can directly be observed in the variation of the response times. More formally, the start time of a stage, as well as its duration may differ from one trial to another. Since each stage is dependent on the previous stages, the variations are expected to increase over time. The applications of the standard SVM methods across trials for each time point ignore the effect of this variation in the stages over the time line. In this section, we use a method to align single trials with each other in order to decrease the variation across the trials for a given time point \( t \).

Single trials are composed of simultaneous recordings from multiple channels. Assuming that there are \( d \) channels, the recorded values at time \( t \) from all channels from an individual trial can be considered as a point \( z(t) = [z_1(t), \ldots, z_d(t)]^T \) in \( \mathbb{R}^d \), and the entire trial as a discrete representation of a 1-dimensional curve in \( \mathbb{R}^d \). A simple measure to find the similarity between these two curves \( z^a \) and \( z^b \) of length \( T \) is to find the sum of the difference vectors for each time point, i.e.,

\[
D(z^a, z^b) = \sum_{t=1}^{T} ||z^a(t) - z^b(t)||^p.
\]

(4–1)

Using dynamic programming, the distance between the two curves can be minimized by stretching these curves to get a better match. which implies that some points may be deleted or gaps may be inserted in either curve. Dynamic Time Warping (DTW) is one of the available matching methods that can be applied on our data [61]. The original algorithm is proposed for one dimensional curves. It is fairly easy to extend the method to \( d \)-dimensions for curves \( z^a \) and \( z^b \) as follows,
\[ DTW(z^a(t), z^b(t)) = \begin{cases} 
DTW(z^a(t-1), z^b(t)) + c_i \\
DTW(z^a(t), z^b(t-1)) + c_d \\
DTW(z^a(t-1), z^b(t-1)) + c_m 
\end{cases} \] (4-2)

where \( c_i \), \( c_d \) and \( c_m \) are the associated costs for insertion, deletion and match. In the current application \( c_i = c_d = 1 \) and \( c_m = 0 \). The matching and its score can be found by backtracking and adding the costs up. The resulting cost is called the edit distance between \( z^a \) and \( z^b \).

Smaller distances between trials mean that the trials are very similar to each other. A subset of similar trials may significantly reduce the variation of the trials with respect to a given time point \( t \). Reduced variation, in return, reduces the noise, and increases the classification accuracy and the significance of the feature selection process.

The edit distance between two trials, which is a similarity measure, can be found using dynamic time warping (DTW). In Figure 4-8, a raster plot of the pair-wise distance scores for 110 single-trial recordings regarding right slanted line stimuli are shown. We sorted all \( 110 \times 110 \) scores in the matrix and then plotted the sorted scores on the right. This plot shows two levels of similarity among trials. We group the trials with scores below 90 and above 90. This pattern is very consistent over all other trials regarding different stimuli types with a threshold between 90 and 100. When these scores are divided into low (0) and high (1) values, it is easy to find similar trials by summing the scores for each trial and sorting the trials by their total score.

A subset of about 35 trials from the top of the list can be considered to be very similar to each other. Therefore, the classification and feature selection algorithms are expected to give better results due to reduced noise. The same method is applied to each different set of trials regarding different stimuli and different sessions to find subsets of similar trials. In
Figure ... the classification and feature selection results are compared between a randomly selected set of trials and the set of similar trials determined by DTW. In Figure 4-9 The improvement after using DTW is very clear for stimulus, categorical and response related differences.

4.5 Selective Support Vector Machines on Neural Data

As explained in Section 4.4, the variation in the stages of the visuomotor task over time precludes using an alignment method to decrease the variation across trials for a given time point. Although the results from the standard kernel-based machine learning methods are intuitive and conclusive, as shown in Chapter 3, better results may be achieved using selective kernel-based machine learning methods. However, one should note that DTW uses the data in the original input space, and works with euclidian distances, whereas similarities between pattern vectors may be better explained in a nonlinear feature space. Selective SVM methods achieve the simultaneous goal of classification in a nonlinear space and alignment of pattern vectors at the same time. We show in this section that the results obtained with the selective SVM methods dominate the results using standard methods like DTW.

The motivation behind the development of selective SVM is the variation in the underlying processes that are to be distinguished by classification. The basic idea is as follows. Let $S_j$ be the a matrix where each row $i$ of $n$ rows is a vector of recordings from trial $i$ at time point, $j$. Now consider the appended matrix $|S_t^T S_{j+1}^T \ldots S_{j+t-1}^T|$, which involves $n \times t$ rows of vector patterns, such that the corresponding rows of $S_j$ and $S_{j+1}$ are two consecutive recordings over the time line from the same trial. This matrix is composed of all pattern vectors within a window of size $t$ from $j$ to $j + t - 1$. Our objective is to provide SVM classifier with more pattern vectors to increase the chance of better point selection. However, the classification algorithm will be effected by the noisy points. We implicitly assume that one point of $t$ from each trial will be allowed to be chosen. In fact, the pattern vector chosen is considered to be a good point, which is expected to
appear, say, at the middle of the time window, however some jitter towards each side of the window caused it to appear within a close neighborhood.

Since selecting the best point from each window and each trial is a very hard problem, we use the pooled free slack (PFS) to eliminate the worst points one by one to achieve the desired points from each window. Although this is a heuristic approach at a first glance, it gives very good results in classification and feature selection.

The code for selective SVM is written in CPLEX 9.0 and dynamic library files are produced to be used in accordance with MATLAB environment. Because of the computational limitations of these software, the entire data could not be solved. The data consists of around 4000 trials. The strategy followed is to consider 200 trials at a time (observed limits of the software) with almost equal distribution of positive and negative labels. Selective SVM algorithm is run over a window of 3 pattern vectors, which is equivalent to 10 milliseconds. The pattern vector with the highest slack is eliminated from each set. This is repeated until there is only one pattern vector per each cell. Although each batch of 200 trials are independent of each other, the cumulative set of the selected points are consistent. The classification and feature selection results significantly dominated the improvements obtained from dynamic time warping method. Since selective SVM tries to find better points, there is an increase in accuracy at the part of the recording before the visual stimuli. This increase poses the question whether this is just an overall increase at every time point. If this is the case, we are not actually detecting anything about the progression of the visuomotor task progression over time. However, when we graph the percentage increase, it is clear from Figure 4-10 that for line-diamond separation, we achieved not only 80% accuracy results between time 130 ms and 200 ms, but also, almost twice the amount of the percent increase compared to the rest of the time scale. This confirms that 40-50 ms after stimulus arrival starts a differentiation in the brain regarding to the abstract categorization of the concept of a
line and a diamond. We want to make a note that the previous studies using standard statistical methods failed to detect any difference for the line-diamond separation.

The go-nogo separation is somewhat confusing because the highest percent increase is before the onset of stimulus. However, this is due to the bias introduced in the brain probably from the uneven distribution of go trials which may be performed back to back and thus carrying over the brain state from one trial to the next. However, it can also be observed that such separation decreases by the time stimuli is arrives to the striate cortex. This point on the time line can be considered as a priming point for this artifact after which a jump in the percent increase can be observed. This jump from 200 ms to 260 ms, being too early for the motor response may correspond to the decision to respond or not.

The results on feature selection is also quite significant. In Figure 4-11, the left column is line-diamond, right column is go-nogo, upper row is feature selection results after DTW, lower row is feature selection results after selective SVM. Note that the interaction of channel 8, which shows the motor response is not observed prominently on the line-diamond case. Instead striate and prestriate cortexes contribute to classification for the aforementioned time periods, for which categorical and response related differences can be observed. However, the assumption that the possibility of back-to-back go trials are confirmed by the excited activity on the motor cortex, even before the stimulus onset.

Since selective SVM tries to separate the two classes for every time point there is an average increase of 10% in the accuracy. This increase can be counteracted by creating a baseline. For this approach we randomly assign class labels to pattern vectors so that we can deduce the increase in the accuracy for random data. In Figure 4-12, the improvement is very clear after the baseline approach. It is also worth noting that there is an actual decrease in the baseline where there is a prominent increase in the accuracy. This case, which has not been detected by any previous studies is very prominently detected using selective SVM method and creating a baseline.
Figure 4-4. Classification and channel selection results for GE
Figure 4-5. Classification and channel selection results for LU
Figure 4-6. Classification and channel selection results for TI
Figure 4-7. Brain snapshots
Figure 4-8. a) Uncoded scores matrix b) Sorted scores c) Coded matrix d) Sorted matrix
Figure 4-9. a) comparative classification results b) comparative channel selection results
Figure 4-10. Improvement classification for line-diamond (left) and go-nogo (right)

Figure 4-11. Improvement in the feature selection for line-diamond (left) and go-nogo (right)
Figure 4-12. Baseline approach to emphasize improvement (top), difference between actual improvement and baseline (bottom)
CHAPTER 5
GENERALIZED EIGENVALUE CLASSIFIERS

In this Chapter we discuss a different class of Support Vector Machine classifiers usually referred as Proximal Support Vector Machines, which was popularized by Mangasarian et al. In this family of classifiers, the notion of a separating hyperplane turns into two hyperplanes, each approximating its own class by minimizing the total distance from its pattern vectors while maximizing the total distance from the opposite set of pattern vectors. The advantage of the proximal classifier model is that the underlying minimization problem reduces to an eigenvalue/eigenvector problem, whose solution is easy to find.

Our main contribution in this chapter are three-fold. The first one is an improvement that would reduce the running time of the current method in the literature. The method developed by Mangasarian requires two eigenvalue problems to be solved, one for each of the approximating hyperplane. However, in Section 5.2, we show that solving only one problem would suffice, using a new regularization technique to find both hyperplanes. We show comparative computational results of our method with the current one. In Section 5.3, we discuss an implementation of the proposed method on parallel computers and presents computational results. Considering a dynamic training set, we introduce an incremental version of the proposed method in Section 5.4 that uses a significantly small kernel to reduce computational complexity of classifying new training points, and improve the generalization capability of the classifier. Similarly, computational results are presented for the incremental method.

5.1 Generalized Eigenvalue Classifiers

In this section we are going to switch to a slightly different notation. Now, consider that $A_{n_1 \times d} = [x_1 \ x_2 \ \ldots \ x_{n_1}]^T$ is a matrix formed by all pattern vectors $x_i \in \mathbb{R}^d$ from the first class, and similarly $B_{n_2 \times d}$ is a matrix formed by all pattern vectors from the second class, and the total number of pattern vectors is $n_1 + n_2 = n$. 
Mangasarian et al. [16] proposes to classify these two sets of points $A$ and $B$ using two hyperplanes instead of a single separating hyperplane as in support vector machines. In Mangasarian’s approach each hyperplane is closest to one set of points, and furthest from the other. Let $x^T w - \gamma = 0$ be a hyperplane in $\mathbb{R}^d$. In order to satisfy the previous condition for the points in $A$, the hyperplane for the first class of pattern vectors can be obtained by solving the following optimization problem:

$$\min_{w, \gamma \neq 0} \|Aw - e\gamma\|^2$$  (5–1)

Here, $e$ is a vector of 1’s. The hyperplane for $B$ can be obtained by minimizing the inverse of the objective function in (5–1). Now, let

$$G = [A - e]^T[A - e], \quad H = [B - e]^T[B - e], \quad z = [w^T \gamma]^T,$$  (5–2)

then equation (5–2), becomes:

$$\min_{z \in \mathbb{R}^m} \frac{z^T G z}{z^T H z}.$$  (5–3)

The expression in (5–3) is the Raleigh quotient of the generalized eigenvalue problem $Gx = \lambda Hx$. The stationary points are obtained at and only at the eigenvectors of (5–3), where the value of the objective function is given by the eigenvalues. When $H$ is positive definite, the Raleigh quotient is bounded and it ranges over the interval determined by minimum and maximum eigenvalues [62]. $H$ is positive definite under the assumption that the columns of $[B - e]$ are linearly independent. The inverse of the objective function in (5–3) has the same eigenvectors and reciprocal eigenvalues. Let $z_{\min} = [w_1 \gamma_1]$ and $z_{\max} = [w_2 \gamma_2]$ be the eigenvectors related to the eigenvalues of smallest and largest modulo, respectively. Then $x^T w_1 - \gamma_1 = 0$ is the closest hyperplane to the set of points in $A$ and the furthest from those in $B$ and $x^T w_2 - \gamma_2 = 0$ is the closest hyperplane to the set of points in $B$ and the furthest from those in $A$. This is depicted in the examples shown in Figure 5-1.
A standard technique in SVMs to obtain a greater separability between sets is to embed the pattern vectors into a nonlinear space, via kernel functions. In this work we use the Gaussian kernel,

\[ K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}. \] (5–4)

In (5–4), \(x_i\) and \(x_j\) denote two pattern vectors in the feature space. This technique usually gives better results, as shown in several studies. Results regarding nonlinearly separable problems \([63, 64]\) still hold and a formulation for the eigenvalues problem can easily be derived. This formulation is given in the next section.

5.2 A New Regularized Fast Classification Method

Recall that \(A\) and \(B\) are the matrices containing the two classes of training points, with each row representing a point in the future space. Let \(G\) and \(H\) be defined as in (5–2). Note that even if \(A\) and \(B\) are full rank, matrices \(G\) and \(H\) are always rank-deficient. The reason is that \(G\) and \(H\) are matrices of order \(d + 1\), and their rank can be at most \(d\). The added complexity due to singularity of the matrices means that special care has to be given to the solution of the generalized eigenvalue problem. Indeed, if the null spaces of \(G\) and \(H\) have a nontrivial intersection, i.e. \(\text{Ker}(A) \cap \text{Ker}(B) \neq 0\), then
the problem is singular and a regularization technique is needed to solve the eigenvalue problem.

Mangasarian et al. proposes to use Tikhonov regularization applied to a two-fold problem:

\[
\begin{align*}
\min_{w, \gamma \neq 0} & \quad \frac{\|Aw - e\gamma\|^2 + \delta\|z\|^2}{\|Bw - e\gamma\|^2}, \\
\text{and} & \\
\min_{w, \gamma \neq 0} & \quad \frac{\|Bw - e\gamma\|^2 + \delta\|z\|^2}{\|Aw - e\gamma\|^2},
\end{align*}
\]

(5–5)

and

(5–6)

5.2.1 The New Regularization Method

where \( \delta \) is the regularization parameter and the new problems are still convex. The minimum eigenvalues-eigenvectors of these problems are approximations of the minimum and the maximum eigenvalues-eigenvectors of equation (5–3). The solutions \( (w_i, \gamma_i), i = 1, 2 \) to (5–5) and (5–6) represent the two hyperplanes approximating the two classes of training points.

In practice, if \( \beta G - \alpha H \) is nonsingular for every \( \alpha \) and \( \beta \), it is possible to transform the problem into another problem that is nonsingular and that has the same eigenvectors of the initial one. We start with the following theorem [65]

**Theorem 5.2.1.** Consider the generalized eigenvalue problem \( Gx = \lambda Hx \) and the transformed \( G^*x = \lambda H^*x \) defined by:

\[
G^* = \tau_1 G - \delta_1 H, \quad H^* = \tau_2 H - \delta_2 G,
\]

(5–7)

for each choice of scalars \( \tau_1, \tau_2, \delta_1 \) and \( \delta_2 \), such that the 2 \( \times \) 2 matrix

\[
\Omega = \begin{pmatrix} \tau_2 & \delta_1 \\ \delta_2 & \tau_1 \end{pmatrix}
\]

(5–8)
is nonsingular. Then the problem $G^*x = \lambda H^*x$ has the same eigenvectors of the problem $Gx = \lambda Hx$. An associated eigenvalue $\lambda^*$ of the transformed problem is related to an eigenvalue $\lambda$ of the original problem by

$$
\lambda = \frac{\tau_2 \lambda^* + \delta_1}{\tau_1 + \delta_2 \lambda^*}.
$$

In the linear case, Theorem 5.2.1 can be applied. By setting $\tau_1 = \tau_2 = 1$ and $\hat{\delta}_1 = -\delta_1, \hat{\delta}_2 = -\delta_2$, the regularized problem becomes

$$
\min_{w, \gamma \neq 0} \|Aw - e^\gamma\|^2 + \hat{\delta}_1 \|Bw - e^\gamma\|^2
\|Bw - e^\gamma\|^2 + \hat{\delta}_2 \|Aw - e^\gamma\|^2.
$$

(5–9)

If $\hat{\delta}_1$ and $\hat{\delta}_2$ are non-negative, $\Omega$ is non-degenerate. The spectrum is now shifted and inverted so that the minimum eigenvalue of the original problem becomes the maximum of the regularized one, and the maximum becomes the minimum eigenvalue. Choosing the eigenvectors related to the new minimum and maximum eigenvalue, we still obtain the same ones of the original problem.

This regularization works for the linear case if we suppose that in each class of the training set there is a number of linearly independent rows that is at least equal to the number of the features. This is often the case and, since the number of points in the training set is much greater than the number of features, $\text{Ker}(G)$ and $\text{Ker}(H)$ have both dimension 1. In this case, the probability of a nontrivial intersection is zero.

In the nonlinear case the situation is different. Using the kernel function (5–4), each element of the kernel matrix is

$$
K(A, B)_{i,j} = e^{-\frac{|A_i - B_j|^2}{\sigma}}.
$$

(5–10)

Let
\[ C = \begin{bmatrix} A \\ B \end{bmatrix}, \]

then, problem (5–1) becomes:

\[
\min_{u, \gamma \neq 0} \| K(A, C)u - e\gamma \|^2 \| K(B, C)u - e\gamma \|^2. \tag{5–11}
\]

Now, the associated eigenvalue problem has matrices of order \( n + 1 \) and rank at most \( d \).
This means a regularization technique is needed, since the problem can be singular.
We propose to generate the following two proximal surfaces:

\[
K(x, C)u_1 - \gamma_1 = 0, \quad K(x, C)u_2 - \gamma_2 = 0 \tag{5–12}
\]

by solving the following problem

\[
\min_{u, \gamma \neq 0} \| K(A, C)u - e\gamma \|^2 + \delta \| \tilde{K}_B u - e\gamma \|^2 \| K(B, C)u - e\gamma \|^2 + \delta \| K_A u - e\gamma \|^2 \tag{5–13}
\]

where \( \tilde{K}_A \) and \( \tilde{K}_B \) are diagonal matrices with the diagonal entries from the matrices \( K(A, C) \) and \( K(B, C) \).
The perturbation theory of eigenvalue problems \[66] provides an estimation of the distance between the original and the regularized eigenvectors. If we call \( z \) an eigenvector of the initial problem and \( z(\delta) \) the corresponding one in the regularized problem, then \( |z - z(\delta)| = O(\delta) \), which means their closeness is in the order of \( \delta \).

As mentioned in the previous section, the minimum and the maximum eigenvalues obtained from the solution of (5–13) provide the proximal planes \( P_i, i = 1, 2 \) to classify the new points. A point \( x \) is classified using the distance

\[
\text{dist}(x, P_i) = \frac{\| K(x, C)u - \gamma \|^2}{\| u \|^2}. \tag{5–14}
\]

and the class of a point \( x \) is determined as

\[
\text{class}(x) = \arg\min_{i=1,2} \{ \text{dist}(x, P_i) \}. \tag{5–15}
\]
Let \( A \in \mathbb{R}^{n_1 \times d} \) and \( B \in \mathbb{R}^{n_2 \times d} \) be the training points in each class. Choose appropriate \( \delta_1, \delta_2 \in \mathbb{R} \) and \( \sigma \).

\[
% Build G and H matrices \\
g = \begin{bmatrix} K(A, C, \sigma), -\text{ones}(n_1, 1) \end{bmatrix}; \nonumber \\
h = \begin{bmatrix} \text{kernel}(B, C, \sigma), -\text{ones}(n_2, 1) \end{bmatrix}; \nonumber \\
G = g^T \ast g; \nonumber \\
H = h^T \ast h; \nonumber \\
% Regularize the problem \\
G^* = G + \delta_1 \ast \text{diag}(H); \nonumber \\
H^* = H + \delta_2 \ast \text{diag}(G); \nonumber \\
% Compute the classification hyperplanes \\
[V, D] = \text{eig}(G^*, H^*); \nonumber
\]

Figure 5-2. ReGEC algorithm

The proposed classification method is outlined in Figure 5-2. Here, \( K(A, C, \sigma) \) is the kernel matrix whose entries are \( K(i, j) = \exp\left\{-\left(||a_i - c_j||^2/\sigma\right)\right\} \), where \( a_i \) is the \( i \)th row of matrix \( A \) and \( c_j \) is the \( j \)th row of matrix \( C \), and \( \sigma \) is the shape parameter of the kernel. Function \( \text{ones}(n_{row}, n_{col}) \) is a matrix of size \( n_{row} \times n_{col} \) with all entries 1, and \( \text{diag}(...) \) returns the main diagonal of a square matrix.

### 5.2.2 Computational Results

The aforementioned methods have been tested on benchmark data sets publicly available. Results regard their performance in terms of classification accuracy and execution time. We used data from different repositories: UCI repository [67], Odewahn et al. [68], and IDA repository [69]. These repositories are widely used to compare the performance of new algorithms to the existing methods. The results regarding the linear kernel have been obtained using the first two repositories. The third one has been used in the non-linear kernel implementation. For each data set, the latter repository offers 100 predefined random splits into training and test sets. For several algorithms, results
Table 5-1. Classification accuracy using linear kernel.

<table>
<thead>
<tr>
<th>dataset</th>
<th>n+k</th>
<th>dim</th>
<th>ReGEC</th>
<th>GEPSVM</th>
<th>SVMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDC</td>
<td>300</td>
<td>7</td>
<td>87.60</td>
<td>86.70</td>
<td>89.00</td>
</tr>
<tr>
<td>Cleveland Heart</td>
<td>297</td>
<td>13</td>
<td>86.05</td>
<td>81.80</td>
<td>83.60</td>
</tr>
<tr>
<td>Pima Indians</td>
<td>768</td>
<td>8</td>
<td>74.91</td>
<td>73.60</td>
<td>75.70</td>
</tr>
<tr>
<td>Galaxy Bright</td>
<td>2462</td>
<td>14</td>
<td>98.24</td>
<td>98.60</td>
<td>98.30</td>
</tr>
</tbody>
</table>

obtained from each trial, including SVMs, are recorded. The accuracy results for the linear
kernel SVMs and GEPSVM are taken from Mangasarian et al. [16] and for the non linear
kernel from [69]. Execution times and the other accuracy results have been calculated
using an Intel Xeon CPU 3.20GHz, 6GB RAM running Red Hat Enterprise Linux WS
release 3 with Matlab 6.5, during normal daylight operations. Matlab function \texttt{eig}
for the solution of the generalized eigenvalue problem has been used for GEPSVM and ReGEC.
The latest releases for LIBSVM [15] and SVMlight [70] have been used to compare these
methods with SVMs.

In tables 5-1 and 5-2, classification accuracy using linear and gaussian kernels have
been evaluated. Tables columns represent: data set name, the number of elements in the
training set (n+k), the number of elements in the test set and the accuracy results for
ReGEC, GEPSVM and SVMs. In Table 5-1, the accuracy results have been evaluated
using ten fold cross validation. In table 5-2, the random splits of IDA repository have
been used. In the linear case comparable accuracy results have been obtained by the three
methods. Using the gaussian kernel, ReGEC and GEPSVM show similar behavior yielding
always results slightly lower than SVMs.

In Tables 5-3 and 5-4, elapsed time is reported. In the linear case ReGEC and GEPSVM
outperform SVMs implementations (LIBSVM and SVM light) in all cases. Furthermore
ReGEC is at least twice faster then GEPSVM. When the gaussian kernel is used, SVMs
implementations achieve better performances with respect to the eigenvalues based
methods. In all cases, ReGEC is faster than GEPSVM.
### Table 5-2. Classification accuracy using gaussian kernel.

<table>
<thead>
<tr>
<th>dataset</th>
<th>n+k</th>
<th>test</th>
<th>m</th>
<th>δ</th>
<th>σ</th>
<th>ReGEC</th>
<th>GEPSVM</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast-cancer</td>
<td>200</td>
<td>77</td>
<td>9</td>
<td>1.e-03</td>
<td>50</td>
<td>73.40</td>
<td>71.73</td>
<td>73.49</td>
</tr>
<tr>
<td>Diabetis</td>
<td>468</td>
<td>300</td>
<td>8</td>
<td>1.e-03</td>
<td>500</td>
<td>74.56</td>
<td>74.75</td>
<td>76.21</td>
</tr>
<tr>
<td>German</td>
<td>700</td>
<td>300</td>
<td>20</td>
<td>1.e-03</td>
<td>500</td>
<td>70.26</td>
<td>69.36</td>
<td>75.66</td>
</tr>
<tr>
<td>Thyroid</td>
<td>140</td>
<td>75</td>
<td>5</td>
<td>1.e-03</td>
<td>0.8</td>
<td>92.76</td>
<td>92.71</td>
<td>95.20</td>
</tr>
<tr>
<td>Heart</td>
<td>170</td>
<td>100</td>
<td>13</td>
<td>1.e-03</td>
<td>120</td>
<td>82.06</td>
<td>81.43</td>
<td>83.05</td>
</tr>
<tr>
<td>Waveform</td>
<td>400</td>
<td>4600</td>
<td>21</td>
<td>1.e-03</td>
<td>150</td>
<td>88.56</td>
<td>87.70</td>
<td>90.21</td>
</tr>
<tr>
<td>Flare-solar</td>
<td>666</td>
<td>400</td>
<td>9</td>
<td>1.e-03</td>
<td>3</td>
<td>58.23</td>
<td>59.63</td>
<td>65.80</td>
</tr>
<tr>
<td>Titanic</td>
<td>150</td>
<td>2051</td>
<td>3</td>
<td>1.e-03</td>
<td>150</td>
<td>75.29</td>
<td>75.77</td>
<td>77.36</td>
</tr>
<tr>
<td>Banana</td>
<td>400</td>
<td>4900</td>
<td>2</td>
<td>1.e-05</td>
<td>0.2</td>
<td>84.44</td>
<td>85.53</td>
<td>89.15</td>
</tr>
</tbody>
</table>

### Table 5-3. Elapsed time in seconds using linear kernel.

<table>
<thead>
<tr>
<th>dataset</th>
<th>ReGEC</th>
<th>GEPSVM</th>
<th>LIBSVM</th>
<th>SVM</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDC</td>
<td>0.1e-03</td>
<td>0.2e-03</td>
<td>0.8991</td>
<td>22.0020</td>
<td></td>
</tr>
<tr>
<td>Cleveland Heart</td>
<td>1.92e-04</td>
<td>3.58e-04</td>
<td>0.0099</td>
<td>0.3801</td>
<td></td>
</tr>
<tr>
<td>Pima Indians</td>
<td>1.21e-04</td>
<td>2.36e-04</td>
<td>15.8737</td>
<td>48.8092</td>
<td></td>
</tr>
<tr>
<td>Galaxy Bright</td>
<td>0.3e-3</td>
<td>0.5e-3</td>
<td>1.2027</td>
<td>21.1280</td>
<td></td>
</tr>
</tbody>
</table>

### Table 5-4. Elapsed time in seconds using gaussian kernel.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ReGEC</th>
<th>GEPSVM</th>
<th>LIBSVM</th>
<th>SVM</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast-cancer</td>
<td>0.0698</td>
<td>0.3545</td>
<td>0.0229</td>
<td>0.1188</td>
<td></td>
</tr>
<tr>
<td>Diabetis</td>
<td>1.1474</td>
<td>5.8743</td>
<td>0.1323</td>
<td>0.2022</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td>3.8177</td>
<td>25.2349</td>
<td>0.2855</td>
<td>0.4005</td>
<td></td>
</tr>
<tr>
<td>Thyroid</td>
<td>0.0243</td>
<td>0.1208</td>
<td>0.0053</td>
<td>0.0781</td>
<td></td>
</tr>
<tr>
<td>Heart</td>
<td>0.0316</td>
<td>0.2139</td>
<td>0.0172</td>
<td>0.1372</td>
<td></td>
</tr>
<tr>
<td>Waveform</td>
<td>0.5962</td>
<td>4.4090</td>
<td>0.0916</td>
<td>0.2228</td>
<td></td>
</tr>
<tr>
<td>Flare-solar</td>
<td>1.8737</td>
<td>16.2658</td>
<td>0.1429</td>
<td>4.4524</td>
<td></td>
</tr>
<tr>
<td>Titanic</td>
<td>0.0269</td>
<td>0.1134</td>
<td>0.0032</td>
<td>7.1953</td>
<td></td>
</tr>
<tr>
<td>Banana</td>
<td>0.4989</td>
<td>3.1102</td>
<td>0.0344</td>
<td>1.3505</td>
<td></td>
</tr>
</tbody>
</table>

93
Finally, a graphical representation of the classification surfaces obtained by ReGEC, GEPSVM and SVMs is given in figure 5-3 relatively to Banana dataset. The three methods show similar class regions. SVMs obtain smoother borders and more regular regions. These differences depend upon the fact that in SVMs the surfaces are characterized by the support vectors and the penalties terms, while in the eigenvalues methods all the points contribute to the solution surfaces. This behavior depends on the fact that eigenvalues methods always maximize the classification accuracy on the training set with respect to kernel and regularization parameters.

5.3 A Parallel Implementation of the Fast Classification Method

Our aim has been to realize an efficient, portable and scalable parallel implementation of ReGEC to be used on different MIMD distributed memory architectures. As well known, these are multiprocessor computers, in which each node has local memory and communicates with the others through message passing. Let us suppose that each processor executes the same program and the same operations on different data (SPMD). Given the algorithm structure, a flexible connection topology is supposed to exist among the nodes, that is, point-to-point communications are allowed, as well as the broadcast and gather of data. Finally, we suppose to have a network in which the processors are in a mesh topology. With this environment in mind, it is natural to develop a program in terms of loosely synchronous processes, executing the same operations on different data,
and synchronizing each other through message passing. To clarify the exposition, we suppose that each node is driven by a single process.

5.3.1 Implementation Details

In Figure 5-2, linear algebra operations are essentially matrix-matrix multiplications and a generalized eigenvalue problem solution. In order to obtain an efficient, portable and scalable parallel implementation of ReGEC we decided to use standard message passing libraries, i.e. BLACS and MPI, and *de facto* standard numerical linear algebra software, PBLAS and ScaLAPACK. Since matrices involved in the algorithm are distributed among processing nodes, memory is used efficiently and no replication of data occurs. On single node, the use of optimized level 3 BLAS and LAPACK routines enables both its efficient use and a favorable computation/communication ratio.

The main routine of PBLAS used in the implementation of Figure 5-2 is PDGEMM to evaluate matrix-matrix multiplications. The current model implementation of the PBLAS assumes the matrix operands to be distributed according to the block scatter decomposition of PBLAS and ScaLAPACK. Routines for eigenvalues problems are not included in PBLAS, but they are covered by ScaLAPACK. The evaluation of the generalized eigenvalue problem $G\mathbf{x} = \lambda H\mathbf{x}$ then performed by using the routine PDSYGVX. We required machine precision in the computation of eigenvalues and, dynamically allocated memory for reorthogonalization of eigenvectors. Current version of ScaLAPACK does not permit to reorthogonalize eigenvectors against those in different processors memory, which can lead to slightly different results, with respect to sequential computation.

We developed the auxiliary routines for parallel kernel computation, and for diagonal matrices operations. Parallel kernel routine is derived by the distribution routine PDMATDIS implemented in HPEC, which loads matrices from files and distributes to processors, accordingly to the block scattered decomposition. It permits to appropriately
load the matrices $A$ and $B$ and to evaluate the elements of the kernel matrix needed by each process.

Finally, the operation count of parallel ReGEC is exactly the same as the sequential one. Thanks to computational characteristics of linear algebra kernels, the parallel implementation of the algorithm described in Figure 5-2 has a computational complexity on $p$ nodes that is exactly $1/p$ of the sequential one, and a communication complexity of one order magnitude less than computational one. This is usually a target in the implementation of parallel linear algebra kernels, because it assures scalable implementations.

5.3.2 Computational Results

The dataset used in this study consists of the genomic sequences of Translation Initiation Site (TIS), which is publicly available [71]. The prediction of TIS in a genomic sequence is an important issue in biological research. This problem can be stated as a classification problem and, although some techniques exist, there is a great potential for the improvement of the accuracy and speed of these methods. Moreover, it provides a significant case study for the analysis of genomic sequences. The aforementioned method has been tested on benchmark data sets obtained from the TIS. Results regard performance in terms of execution time and efficiency. Execution times and the other accuracy results have been calculated using a Beowulf cluster of 16 Pentium 4 1.5 GHz, with 512MB RAM, connected with a Fast Ethernet network. Each node runs a Linux kernel 2.4.20, gcc compiler 2.96, mpich 1.2.5, BLACS 1.1, ScaLAPACK 1.7, LAPACK 3.0, BLAS with ATLAS optimization. Tests have been performed on idle workstations; the time refers to wall clock time of the slower executing node and it has been measured with function MPI_WTIME() provided by mpich. The maximum memory available on each node led to the impossibility to run some test cases on a small number of processors.

The execution times and parallel efficiency are shown in Tables 5-5 and 5-6, using different number of either training elements and CPU. Tests have been performed on
logical 2D meshes of 1(1), 2(1 × 2), 4 (2 × 2), 8(2 × 4) and 16(4 × 4) processors. The training sets have dimensions ranging between 500 and 9000 points.

Table 5-5. Execution times for the parallel implementation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>2.99</td>
<td>3.59</td>
<td>3.07</td>
<td>3.51</td>
<td>4.00</td>
</tr>
<tr>
<td>1000</td>
<td>21.90</td>
<td>17.79</td>
<td>12.29</td>
<td>12.61</td>
<td>12.43</td>
</tr>
<tr>
<td>2000</td>
<td>162.12</td>
<td>89.79</td>
<td>55.95</td>
<td>46.59</td>
<td>40.54</td>
</tr>
<tr>
<td>3000</td>
<td>532.42</td>
<td>260.39</td>
<td>143.93</td>
<td>109.63</td>
<td>87.30</td>
</tr>
<tr>
<td>4000</td>
<td>1487.87</td>
<td>562.70</td>
<td>290.02</td>
<td>205.95</td>
<td>155.39</td>
</tr>
<tr>
<td>5000</td>
<td>2887.51</td>
<td>1050.02</td>
<td>265.92</td>
<td>342.22</td>
<td>247.36</td>
</tr>
<tr>
<td>6000</td>
<td>-</td>
<td>1921.13</td>
<td>812.64</td>
<td>523.99</td>
<td>365.92</td>
</tr>
<tr>
<td>7000</td>
<td>-</td>
<td>3414.97</td>
<td>1298.75</td>
<td>753.63</td>
<td>514.66</td>
</tr>
<tr>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>1875.02</td>
<td>1046.08</td>
<td>693.84</td>
</tr>
<tr>
<td>9000</td>
<td>-</td>
<td>-</td>
<td>2733.95</td>
<td>1421.28</td>
<td>913.16</td>
</tr>
</tbody>
</table>

In table 5-6 the efficiency is calculated using the following formula:

$$ eff = \frac{t_1}{#cpu \times t_{#cpu}} $$  \hspace{1cm} (5-16)

where \( t_\# \) is the execution time using \# number of cpu. In all cases for which we could not evaluate sequential or parallel execution time on a small number of nodes, we set efficiency to 1 on the minimum number of processors on which we could run the application.

Table 5-6. Efficiency for the parallel implementation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1</td>
<td>0.4175</td>
<td>0.2442</td>
<td>0.1066</td>
<td>0.0468</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>0.6157</td>
<td>0.4458</td>
<td>0.2172</td>
<td>0.1102</td>
</tr>
<tr>
<td>2000</td>
<td>1</td>
<td>0.9027</td>
<td>0.7244</td>
<td>0.4349</td>
<td>0.2499</td>
</tr>
<tr>
<td>3000</td>
<td>1</td>
<td>1.0223</td>
<td>0.9248</td>
<td>0.6071</td>
<td>0.3812</td>
</tr>
<tr>
<td>4000</td>
<td>1</td>
<td>1.3221</td>
<td>1.2825</td>
<td>0.9031</td>
<td>0.5984</td>
</tr>
<tr>
<td>5000</td>
<td>1</td>
<td>1.3750</td>
<td>2.7146</td>
<td>1.0547</td>
<td>0.7296</td>
</tr>
<tr>
<td>6000</td>
<td>-</td>
<td>1</td>
<td>1.1820</td>
<td>0.9166</td>
<td>0.6563</td>
</tr>
<tr>
<td>7000</td>
<td>-</td>
<td>-</td>
<td>1.3147</td>
<td>1.1328</td>
<td>0.8294</td>
</tr>
<tr>
<td>8000</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0.8962</td>
<td>0.6756</td>
</tr>
<tr>
<td>9000</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0.9618</td>
<td>0.7485</td>
</tr>
</tbody>
</table>

Results show that, for an increasing number of processors, the execution time decreases proportionally, if the problem to be solved has sufficient computational
complexity. Moreover, time reduction increases for larger problems, with a consistent gain in performance. We note that, in some cases efficiency is above 1, due to limited memory on each cluster node; nevertheless a sensible execution time reduction is obtained when the number of processors increases. We can conclude that parallel ReGEC is efficient and scalable on the target architecture.

5.4 An Incremental Classification Algorithm

Classification problems may involve a large number of training points. One immediate solution is to select a subset of points that would retain the characteristics of the training set. A second problem arises when a new training data point becomes available for training. A desirable method as a solution to the second problem should be based on an efficient evaluation of how the new point may influence the classification function, rather than a complete training of the incrementally augmented training set.

Datasets in almost every application area are ever growing and are continuously updated. Moreover, numerous applications on massive datasets are emerging \([44]\), which require efficient computational procedures to respond to the dynamics of large databases. As machine learning becomes a part of data intensive computation systems, updating the learning system becomes intractable in many cases. Therefore, incremental methods that require some minimal computational burden are strongly preferred. For this purpose several methods, especially in the kernel-based nonlinear classification cases, have been proposed to reduce the size of the training set, and thus, the related kernel \([45–49]\). All of these methods show that a sensible data reduction is possible while maintaining a comparable level of classification accuracy.

In this study, a new method that finds a small subset of the training dataset is introduced. The amount of reduction in the training set can be as large as 98% with comparable classification accuracy and improved consistency with respect to the original training set. The proposed subset selection method starts with an initial set of points and incrementally expands this set by adding those points which contribute to improving
classification accuracy. The main idea is to use the small subset of points to solve the general eigenvalue problem, and therefore the evaluation of the contributions for new points is performed in conjunction with ReGEC. Thus, we refer to our method as Incremental ReGEC (I-ReGEC).

5.4.1 Incremental Subset Selection Algorithm

The dimension of generalized eigenvalue problem (5–13) is equal to \( n \), the number of points in the training set, plus 1. Since the computational complexity of the operation is in the order of \( O((n + k)^3) \), it is important to develop methods that are capable of finding a small and robust set of points that retains the characteristics of the entire training set and provides comparable accuracy results. A kernel built from a smaller subset is computationally more efficient in predicting new points compared to kernels that use the entire training set. Furthermore, a smaller set of points reduces the probability of over-fitting. Finally, as new points become available, the cost of retraining the algorithm decreases if the influence of the new points on the classification function is only evaluated by the small subset, rather than the whole training set. The main idea is to exploit the efficiency of solving a small eigenvalue problem. Therefore, we use ReGEC as the internal method to evaluate the classification accuracy on the entire training set.

The algorithm takes an initial set of points \( C^0 \) and the entire training set \( C \) as input, such that \( C \supset C_0 = A_0 \cup B_0 \), and \( A_0 \) and \( B_0 \) are sets of points in \( C_0 \) that belong to the two classes \( A \) and \( B \). We refer to \( C_0 \) as the incremental subset. Let \( \Gamma_0 = C \setminus C_0 \) be the initial set of points that can be included in the incremental subset. ReGEC classifies all of the points in the training set \( C \) using the kernel from \( C_0 \). Let \( P_{A_0} \) and \( P_{B_0} \) be the hyperplanes found by ReGEC, \( r_0 \) be the classification accuracy and \( M_0 \) be the points that are misclassified. Then, among the points in \( \Gamma_0 \cap M_0 \) the point that is farthest from its respective hyperplane is selected, i.e.

\[
x_1 = x_i : \max_{x \in \{\Gamma_0 \cap M_0\}} \left\{ dist(x, P_{\text{class}(x)}) \right\}
\]

(5–17)
where \(\text{class}(x)\) returns \(A\) or \(B\) depending on the class of \(x\). This point is the candidate point to be included in the incremental subset. This choice is based on the idea that a point very far from its plane may be needed in the classification subset in order to improve accuracy. We update the incremental set as \(C_1 = C_0 \cup \{x_1\}\). Then, we classify the entire training set \(C\) using the points in \(C_1\) to build the kernel. Let the classification accuracy be \(r_1\). If \(r_1 > r_0\) then we keep the new subset; otherwise we reject the new point, that is \(C_1 = C_0\). In both cases \(\Gamma_1 = \Gamma_0 \setminus \{x_1\}\). The algorithm repeats until the condition \(|\Gamma_k| = 0\) is reached at some iteration \(k\). The algorithm can be summarized as follows:

\begin{algorithm}
\caption{I-ReGEC(\(C_0, C\))}
\begin{algorithmic}[1]
\State \(\Gamma_0 = C \setminus C_0\)
\State \(\{r_0, M_0\} = \text{Classify}(C, C_0)\)
\State \(k = 1\)
\While{\(|\Gamma_k| > 0\)}
\State \(x_k = \min_{x \in \{M_k \cap \Gamma_{k-1}\}} \{\text{dist}(x, P_{\text{class}(x)})\}\)
\State \(\{r_k, M_k\} = \text{Classify}(C, \{C_{k-1} \cup \{x_k\}\})\)
\If{\(r_k > r_{k-1}\)}
\State \(C_k = C_{k-1} \cup \{x_k\}\)
\State \(\Gamma_k = \Gamma_{k-1} \setminus \{x_k\}\)
\State \(k = k + 1\)
\EndIf
\EndWhile
\end{algorithmic}
\end{algorithm}

In Figure 5-4 a graphical example of this approach is shown. The classification surfaces of the two classes (dark and white), generated using 400 training points of the Banana dataset [69], clearly define the aim of our strategy. Indeed, when the ReGEC algorithm is trained on all of the training points the classification boundaries are significantly affected by noisy points (left). On the other hand, I-ReGEC method achieves clearly defined boundaries (right). Furthermore, the number of points needed in the example to generate the classification hyperplane are only 23 in I-ReGEC compared to 400 points in ReGEC.
5.4.2 Initial Points Selection

In the Section 5.4.1, we assumed that we have a starting set of points for I-ReGEC. However, we have not mentioned the bias this initial set introduces. Since the initial points permanently become a part of the incremental subset, it is intuitive that such points should be chosen carefully. In this section we show how the initial set of points influence the performance of the incremental selection algorithm. Clustering techniques can be adapted to obtain better data representations [72]. For this purpose, we compare $k$ randomly selected starting points for each class, and a set of points determined by a simple $k$-means method [73], also for each class. We show that it is possible to reach higher classification accuracy and a more consistent representation of the training set using k-means method.

The two datasets used for the comparison have 2 dimensions, in order to show the consistency of the k-means method over random selection, graphically. From each class, $k$ points are chosen for both random and k-means methods. The first dataset is the Banana dataset with 400 training points and 4900 test points. The second set of points is the Chessboard dataset. It contains 16 squares, with a total of 1000 training and 5400 test points.

First, classification parameters are determined using a ten-fold cross-validation using the training and test points. An initial set of starting points is chosen randomly, and
b) using the barycenters of the clusters produced by the k-means method. Each set is used as input to I-ReGEC algorithm, which returns a final incremental subset of points $C^*$, and the final classification accuracy. Using the same parameters we repeat the procedure of choosing initial points and running I-ReGEC 100 times for both the random and the k-means methods as the generator of the initial sets. Let $C^*_i$ be the final subset of points produced in the $t^{th}$ repetition. Then, for each kernel produced by $C_i$, we classify a dense set of evenly distributed points in the rectangle that encloses the entire dataset. Let $\mathbf{x}$ be one of such points in the rectangle and $y_i \in \{-1, 1\}$ be the classification result using the kernel-based on $C_i$. Then the value $\hat{y} = \left| \sum_{i=1}^{100} y_i \right|/100$ is an estimator of the probability that $\mathbf{x}$ is always classified in the same class. We can say that the closer $\hat{y}$ is to 1, the more consistently it is classified.

In Figure 5-5, white color is associated to the points for which $\hat{y} = 1$ and black for $\hat{y} = 0.5$. The lighter regions are more consistent compared to dark regions, where the points have the same probability to be classified in one of the two classes. The influence of the starting points on the resulting classification can be seen clearly. The Banana dataset has few clusters of data and consequently, for a choice of $k = 5$, the average classification accuracy slightly changes between random initial points, which produce a classification accuracy of 84.5%, and k-means initial points, with accuracy of 85.5%. In order to compare the consistency of the two initial points selection strategies, we measure the standard deviation of the $\hat{y}$ values for the points in the rectangle. The k-means method achieves a standard deviation of 0.01 compared to the standard deviation of 0.05 from the random method, which means that k-means method has a higher classification consistency than random selection.

For the Chessboard dataset, the clusters are clearly separated for each class when $k = 8$. The difference is more pronounced both in terms of classification accuracy and consistency. Random selection of initial points could only reach a classification accuracy
of 72.1 %, whereas k-means reaches 97.6 % accuracy. The difference in classification consistency is far more evident compared to the Banana dataset, with a standard deviation of 1.45 for random selection and 0.04 for k-means. We can empirically infer from the results that a knowledge regarding the dataset and the choice of initial points influences both classification accuracy and classification consistency. This influence may be greater as the number of clusters increases.

We also investigated the effect of the number of initial points $k$ for each class using the k-means method on the Chessboard dataset. In Figure 5-6, the graph on top is the classification accuracy versus the total number of initial points $2k$ from both classes. It reaches its peak at 16 (for $k = 8$), after which it slightly decreases and continues at a steady state of accuracy for higher values of $k$. This result empirically shows that there is a minimum $k$, with which we reach high accuracy results. Although the decrease in the accuracy is not significant for larger values of $k$, the kernel to be used in I-ReGEC
unnecessarily increases. This is shown by the bottom graph in Figure 5-6 which shows the number of points selected by I-ReGEC versus the number of initial points. Again, no additional points are added to the initial 16 (for \( k = 8 \)), and the number of points added are almost the same beyond. This means that the initial set of points reaches a minimum at an ideal number of \( k \) and it grows linearly with \( k \). One simple and practical way of finding a good \( k \) is to increase \( k \) incrementally and detecting the lowest value of \( k \) with higher classification accuracy.

![Graph showing performance of I-ReGEC with respect to the number of starting points](image)

Figure 5-6. Performance of I-ReGEC with respect to the number of starting points

### 5.4.3 Computational Results

I-ReGEC is tested on publicly available benchmark data sets. Results regarding its performance in terms of classification accuracy are presented. We used data from IDA [69] and from UCI [67] repositories, both of which are widely used to compare the performance of new algorithms to existing methods. The accuracy results for the nonlinear kernel are taken from [69]. Accuracy results are calculated using an Intel Xeon CPU 3.20GHz, 6GB RAM running Red Hat Enterprise Linux WS release 3 with Matlab 6.5. Matlab function `eig` for the solution of the generalized eigenvalue problem is used for ReGEC.

In Table 5-7, for each data set, name, dimension of the training and test sets, and the number of features are reported. In Table 5-8, classification accuracy is evaluated using Gaussian kernel for ReGEC, I-ReGEC, and SVM, using ten-fold cross-validation to
Table 5-7. Datasets characteristics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>train</th>
<th>test</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banana</td>
<td>400</td>
<td>4900</td>
<td>2</td>
</tr>
<tr>
<td>German</td>
<td>700</td>
<td>300</td>
<td>20</td>
</tr>
<tr>
<td>Diabetis</td>
<td>468</td>
<td>300</td>
<td>8</td>
</tr>
<tr>
<td>Haberman</td>
<td>275</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>Bupa</td>
<td>310</td>
<td>35</td>
<td>6</td>
</tr>
<tr>
<td>Votes</td>
<td>391</td>
<td>44</td>
<td>16</td>
</tr>
<tr>
<td>WPBC</td>
<td>99</td>
<td>11</td>
<td>32</td>
</tr>
<tr>
<td>Thyroid</td>
<td>140</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td>Flare-solar</td>
<td>666</td>
<td>400</td>
<td>9</td>
</tr>
</tbody>
</table>

determine parameters. A Gaussian kernel is used for each classifier and the value of the best kernel parameter $\sigma$ together with the $k$ value for the k-means method for I-ReGEC are also included in the table. The $k$ value for each dataset is empirically determined as follows: first, the best $\sigma$ value is determined for $k = 2$ using ten-fold cross-validation; then, the best $k$ value is determined by gradually increasing its value.

I-ReGEC is nearly always more accurate than ReGEC. The slight difference in accuracy for the two datasets where ReGEC gives better results could be due to the cross validation procedure. We have also compared the accuracy results of I-ReGEC with SVM. Results are always slightly lower than SVM, except for one data set. The relative difference of accuracy, i.e., the absolute difference of the accuracies of I-ReGEC and SVM, divided by the maximum value, is less then 8.2%, except the case of Flare-solar (11.50%) and Bupa dataset (15.55%).

In Table 5-9 the dimension of incremental datasets and the percentage with respect to the dimension of the training set is given. In all cases, I-ReGEC produced a subset composed of less then 8.85% of the training set with a comparable classification accuracy on the test sets with respect to the original ReGEC method.
Table 5-8. Classification accuracy for ReGEC, I-ReGEC and SVM

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ReGEC</th>
<th>I-ReGEC</th>
<th>SVMs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>σ</td>
<td>acc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>acc</td>
<td></td>
</tr>
<tr>
<td>Banana</td>
<td>400</td>
<td>0.2</td>
<td>84.44</td>
</tr>
<tr>
<td>German</td>
<td>700</td>
<td>50.2</td>
<td>70.26</td>
</tr>
<tr>
<td>Diabetis</td>
<td>468</td>
<td>500</td>
<td>74.56</td>
</tr>
<tr>
<td>Haberman</td>
<td>275</td>
<td>1200</td>
<td>73.26</td>
</tr>
<tr>
<td>Bupa</td>
<td>310</td>
<td>200</td>
<td>59.03</td>
</tr>
<tr>
<td>Votes</td>
<td>391</td>
<td>50</td>
<td>95.09</td>
</tr>
<tr>
<td>WPBC</td>
<td>99</td>
<td>1000</td>
<td>58.36</td>
</tr>
<tr>
<td>Thyroid</td>
<td>140</td>
<td>0.8</td>
<td>92.76</td>
</tr>
<tr>
<td>Flare-solar</td>
<td>666</td>
<td>3</td>
<td>58.23</td>
</tr>
</tbody>
</table>

Table 5-9. Incremental dataset using I-ReGEC and percentage of the training set

<table>
<thead>
<tr>
<th>Dataset</th>
<th>I-ReGEC</th>
<th>chunk</th>
<th>% of train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banana</td>
<td>15.70</td>
<td>3.93</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td>29.09</td>
<td>4.16</td>
<td></td>
</tr>
<tr>
<td>Diabetis</td>
<td>16.63</td>
<td>3.55</td>
<td></td>
</tr>
<tr>
<td>Haberman</td>
<td>7.59</td>
<td>2.76</td>
<td></td>
</tr>
<tr>
<td>Bupa</td>
<td>15.28</td>
<td>4.93</td>
<td></td>
</tr>
<tr>
<td>Votes</td>
<td>25.90</td>
<td>6.62</td>
<td></td>
</tr>
<tr>
<td>WPBC</td>
<td>4.20</td>
<td>4.30</td>
<td></td>
</tr>
<tr>
<td>Thyroid</td>
<td>12.40</td>
<td>8.86</td>
<td></td>
</tr>
<tr>
<td>Flare-solar</td>
<td>9.67</td>
<td>1.45</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 6
CONCLUSION

The aim of this dissertation was to study core problems in recently developed very efficient and successful machine learning methods. The common characteristic for these machine learning methods is that they can accommodate a kernel function which implicitly defines a map from the original input space to a feature space. The mapped feature space can handle nonlinear surfaces for classification and for finding nonlinear regression functions. Aside from the standard support vector classification and support vector regression methods, some other kernel-based feature selection algorithms were also reviewed, such as the adaptive scaling for SVMs, which uses an iterative combination of standard support vector machines and a conjugate gradient technique to find the relative contribution of each feature to classification. The standard methods have been perfected over years and it is hard to come up with improvement methods, but the other algorithms, such as the feature selection method, can be compared with a simpler feature selection to see how effective it is.

Selective support vector machines have been shown on visual examples to work on finding the most separated points out of many points per set. The similar results also apply for regression, although the results are not as prominent as SelSVM. The pattern vectors are chosen as far from each other as possible for different classes, regardless whether the data set is perfectly separable or not, which shows robustness. In fact, for higher values of the band parameter $e(\cdot)$, the distribution takes the shape of a linear classifier. From a theoretical point of view, selective support vector machines, being a novel method of selective classification, have a lot to explore, starting with the efforts to solve the hard selection problem using integer programming and combinatorial optimization techniques. Other kernel-based methods, whose selective versions are not presented here, can be developed based on the same premise. For example, selective proximal support vector machines can be developed easily, studies of which are ongoing. As far as the current relaxation goes, different schemes can be introduced to distribute
slack to different sets. The distance of points from each other can be a starting measure for uneven free slack distribution rule.

We studied the integration of visual and motor cortices in macaque monkeys using kernel-based machine learning algorithms. The main contribution of the study is that the recordings from a large set of cortices involved in the visuomotor task are considered simultaneously using non-linear mapping, as opposed to the previous studies that revolved around methods that study individual channels.

Through classification, we detected the onset times for the autonomous processing of the visual stimuli in the primary visual cortex (V1). We have clearly detected the differences between stimuli which has different alignment (right or left slanted) as each stimuli has a detailed map of the spatial information in vision. In our experimental paradigm, the visual discrimination task involves discrimination of line and diamond stimuli regardless of their alignment. We observed that the discrimination is time-locked to the early stimulus onset in the striate cortex. The response related differences involve motor planning and execution of the response to the GO stimuli. The execution is dependent on the discrimination of the stimulus, and the GO cue for the specific session. We observed that differences in the GO vs NOGO responses is time-locked to the discrimination phase.

The onset and the duration of the stimulus come from separate distributions for each of the visual, categorical and response related stages. The stages are hypothesized to be sequentially dependent. Therefore, at the individual trial level, a delay on the onset or offset of a stage would cause further delays in the subsequent stages. We used dynamic time warping algorithm to find those single trials within each session and stimulus type to reduce variability, which resulted in significant improvements in both classification and channel selection methods. This shows that, with an appropriate method, the single trials can be aligned. Moreover this alignment may offer insights to the variability of the underlying stages of the visual discrimination task.
Selective support vector machines were developed for further improvement on the alignment of single trial time series, such that the alignment of pattern vectors were done within a time window, by picking the points which would result in as large separation of two classes as possible in different stages of the visuomotor task. The results were shown to improve significantly for the line-diamond and go-nogo cases. Although the improvement can be observed for each time point, including the time period before the stimulus, the anticipated time intervals showed an improvement twice in percent values as much compared to the rest of the increase in the accuracy, especially in the line-diamond case. For the go-nogo case, although prestimulus period has one of the highest increase in the accuracy, this result can be discarded since the effect is due to repeated go trials, and the resulting excited state of the brain being detected after this effect. It is also confirmed that after the onset of stimulus, the transfer of the image to the occipital lobe primes the system and causes the accuracy to drop until the end of the line-diamond period, at which time the difference increases. This time of this increase is still early for response generation and implies that it may be involved in deciding whether to respond or not.

The selective SVM study may easily be extended to other macaque monkeys and similar patterns may be searched to understand the integration of the visual and motor cortices in a broader sense. In fact, selective support vector regression can also be used to make a prediction on the response times for go trials over a sliding window. A possible scenario is to compare the results of classification and regression for the same time points and window size.

Selective SVMs can easily be implemented whenever there are repetitive time series to be categorized for each time point. Such applications may involve recordings from epileptic patients whose nervous systems are periodically stimulated. In a sense, the repeated epochs can be treated as single trial data, and dynamics of the brain as a response to this stimulation can be studied to analyze and understand how epileptic seizures develop, and what can be done to stop them. Briefly, any database of time series that are repetitions
of a possibly nonlinear, multidimensional source could be subject to selective kernel-based
learning methods.

Research activities related to supervised learning have an important role in many
scientific and engineering applications. In the present work a novel regularization
technique *regularized generalized eigenvalue classifier* ReGEC and its application has
been proposed and tested against other methods on a number of datasets. Results show
that the proposed method has a classification accuracy comparable to other methods,
has a computational performance comparable to most of the other methods, and is
much faster than the others in the linear case. These advantages are strong motivations
for ReGEC to be used in a parallel computational platforms. Parallel implementation
of this algorithm provide time efficiency and computational accuracy. The proposed
implementation is tested on a large scale genomic data base. The preliminary results
show the proposed implementation to be efficient and scalable. The future work may
include testing this implementation on large scale data sets from different research areas,
mostly in the biomedical domain, and comparing its performance with the other parallel
classification methods.

For further computational advantage without substantial sacrifice on the results,
I-ReGEC, a novel incremental classification technique is introduced, with dramatic results
in reducing the cardinality of training sets, when applied to general eigenvalue classifiers.
The proposed method achieves a high classification consistency and classification accuracy
comparable with other SVM methods. Furthermore, it allows efficient online updating of
the classification function when new training points become available. I-ReGEC method
can be improved by adaptive techniques for the selection of the initial points, in order
to find better strategies to build the incremental subset. Furthermore, new criteria for
including new points to the incremental subset or removing less promising points from the
incremental subset may be considered.
REFERENCES


BIOGRAPHICAL SKETCH

Onur Şeref is a doctoral candidate in industrial and systems engineering at the University of Florida. His research focuses on data mining and optimization methods in biomedical applications. Onur is also interested in problems on network flow optimization.

Onur received his bachelors and masters degrees in industrial engineering from Middle East Technical University (METU) in Ankara, Turkey in July 1998 and July 2001, respectively. He plans to graduate with his Ph.D. degree in December 2006 and pursue a career in academia, where he can continue his research and other academic involvement.