Kernel Methods for High-Dimensional Biological Data

Dissertation

der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Majid M. Beigi
aus Isfahan, Iran

Tübingen

2008

Tag der mündlichen Qualifikation: 4.7.2008
Dekan: Prof. Dr. Michael Diehl
1. Berichterstatter: Prof. Dr. Andreas Zell
2. Berichterstatter: Prof. Dr. Oliver Kohlbacher

Abstract

This thesis addresses the problem of finding robust, fast and precise learning methods for noisy, incomplete high-dimensional biological data by means of so-called kernel methods. Kernel methods are at the heart of many modern machine learning techniques. The intuitive idea behind kernel methods is that the data are nonlinearly mapped into a higher dimensional feature space, related to a nonlinear kernel function. In general, such a procedure has the advantage that inseparable data becomes linearly separable. A kernel function represents a computational trick, which makes it possible to represent linear patterns efficiently in high–dimensional spaces, to ensure adequate representational power.

We begin with a short overview over the basic theoretical concepts, which are necessary to understand the kernel-based algorithms employed in this work. We present some elements of statistical learning and regularization theory and introduce the concept of kernel functions. Then, we derive the Support Vector Machine (SVM) algorithm from Tikhonov regularization and geometric perspectives.

After this we turn our attention to applications of kernel methods for some problems of high-dimensional biological data: To develop an accurate method for the classification of (G-Protein Coupled receptors) GPCRs families, especially at the sub-subfamily level, where we have a low number of protein sequences, we develop a new approach of oversampling, **S**ynthetic **P**rotein **S**equence **O**versampling (SPSO), in which the minority class in the data space is oversampled by creating synthetic protein sequences, considering the distribution of the minor and major classes. SPSO can be used for protein classification problems and remote homology detection, where classifiers must detect a remote relation between unknown sequence and training data with an imbalance problem.

Another problem from neurobiology of bats, which is surveyed in the content of this thesis, is the pattern recognition and classification of biosonar signals by means of kernel methods. We study the classification of biosonar signals as an example of random process signals which contain local similarities. Inspired by the solutions for remote homology detection in protein families, we suggest a similar kernel to string kernels which measures the similarity of two time series. The more time series share similar subsequences, the more similar they are. We also implement a more general kernel, which considers warping in the subsequences. It measures the whole similarities of all warped non contiguous subsequences of the two time series, independent of their positions. Furthermore, having a set of the kernels for similarity extraction in time-series for different sizes of subsequences, we find the optimal linear combination of kernels. We then use those kernels directly in

a SVM-based classifier. The results show that those kernels allow for a very reliable discrimination of reflected sonar echoes from different objects. We also present an algorithm based on gradient boosting for biosonar data classification. We present two kinds of base learners for the gradient boosting: Ordinary Least Squares (OLS) and kernel-based base learners. The main point of the signal preprocessing in our method, for biosonar classification, is using a filter bank like that of the hearing system of bats. With this filter bank, the one-dimensional sonar echoes are converted into shorter length but more informative multi-dimensional signals. We get efficient and accurate results with the newly proposed kernel based boosting method.

As a last application of kernel methods in this thesis, we deal with classification of biological data, having additive unknown noise. Activity profiles in the Forced Swimming Test (FST) for animal behavior classification are examples of those signals. We implement FIR-based classifiers for animal behavior classification in which a Finite Impulse Response (FIR) filter is used to filter out the additive noise from activity profiles. The parameters of the FIR filter are obtained via maximizing the accuracy of a classifier that tries to make a discrimination between two classes of the activity profiles. Our proposed behavior classification method provides a reliable discrimination of different classes of antidepressant drugs (imipramine and desipramine) administered to rats versus a vehicle-treated group.

Zusammenfassung

Diese Disseration befasst sich mit dem Problem, robuste, schnelle und präzise Lernmethoden für verrauschte und unvollständige hochdimensionale biologische Daten durch Kernmethoden (engl. kernel methods) zu finden. Die intuitive Idee hinter diesen Verfahren ist, dass die Daten mithilfe nicht-linearer Kernoperatoren in einen höherdimensionalen Merkmalsraum eingebettet werden. Ein Kernoperator stellt einen rechnerischen Trick dar, der es ermöglicht, lineare Muster wirkungsvoll in hochdimensionalen Räumen abzubilden.

Zu Beginn soll ein kurzer Überblick über die grundlegenden theoretischen Konzepte der kernbasierten Algorithmen gegeben werden, die in dieser Arbeit zur Anwendung kommen. Darin werden auch einige Elemente des statistischen Lernens und der Regularisierungstheorie sowie das Konzept der Kernoperatoren vorgestellt. Anschließend wird die Support-Vektor-Maschine, d. h., der SVM-Algorithmus, aus der Tikhonov-Regulasierung und geometrischen Perspektiven hergeleitet.

Nach dieser Einführung liegt das Augenmerk auf Anwendungen von Kernmethoden für einige Probleme mit hochdimensionalen biologischen Daten: Es wird eine genaue Methode für die Klassifizierung von G-Protein-gekoppelten Rezeptor Familien, (eng. G-Protein Coupled Receptor ‚GPCR), entwickelt, insbesondere für die Unterunterfamilie, in der nur eine geringe Anzahl von Proteinsequenzen zur Verfügung steht. Dazu wird ein neuer Ansatz zur Überabtastung der Stichprobe vorgestellt, die Überabtastung synthetischer Proteinsequenzen (engl. Synthetic Protein Sequence Oversampling, SPSO). Die Minderheitenklasse im Datenraum wird dabei unter Berücksichtigung der Verteilung der Mehr- und Minderheitenklassen übermäßig abgetastet, indem SPSO künstliche Proteinsequenzen erzeugt. SPSO kann für Proteinklassifikationsprobleme und zur Erkennung entfernter Verwandtschaften (engl. remote homology) genutzt werden, wobei Klassifikatoren eine entfernte Homologie zwischen einer unbekannten Sequenz und Trainingsdaten in ungleich verteilten Stichproben erkennen müssen.

Eine weitere Fragestellung, die in dieser Arbeit untersucht wird, stellt die Anwendung von Kernmethoden zur Mustererkennung und Klassifikation von Biosonarsignalen, die in der Neurobiologie der Fledermäuse bedeutsam sind, dar. Die Klassifikation der Biosonarsignale wird als Beispiel für zufällige Prozesssignale, die lokale Ähnlichkeit aufweisen, angeführt.

Inspiriert durch die Ergebnisse für die Detektion entfernter Homologien in Proteinfamilien empfiehlt sich ein den Kernoperatoren für Zeichenketten ähnlicher Kernoperator, der die Ähnlichkeit zweier Zeitreihen misst. Zwei Zeitreihen ähneln sich umso mehr, je

mehr ähnliche Teilsequenzen sie aufweisen. Weiterhin wird ein allgemeinerer Kernoperator implementiert, der Verzerrungen der Teilsequenzen berücksichtigt. Dieser misst die gesamte Ähnlichkeit aller verzerrten, nicht zusammenhängenden Teilsequenzen der beiden Zeitreihen unabhängig von ihren Positionen. Darüber hinaus lässt sich die optimale Linearkombination von Kernoperatoren identifizieren, wenn eine Menge von Kernoperatoren zur Ähnlichkeitsextraktion in Zeitreihen verschiedener Größen von Teilsequenzen zur Verfügung steht.

Diese Kernoperatoren werden dann direkt in einem SVM-basierten Klassifikator verwendet. Die Ergebnisse zeigen, dass diese Kernel eine sehr zuverlässige Unterscheidung reflektierter Sonarechos verschiedener Objekte erlauben. Zur Klassifikation von Biosonardaten wird ein auf Gradientverstärkung (engl. gradient boosting) basierender Klassifikationsalgorithmus vorgestellt. Zwei Arten von Basislernverfahren für die Gradientverstärkung werden angewendet: Die Methode der gewöhnlichen kleinsten Quadrate (engl. Ordinary Least Squares, OLS) und kernbasierte Lernverfahren. Der Schwerpunkt der hier dargelegten Methode zur Signalvorverarbeitung zum Zwecke der Biosonarklassifikation besteht in der Anwendung einer Filterbank in Analogie zum Hörsystem der Fledermäuse.

Mit dieser Filterbank werden die eindimensionalen Sonarechos in ihrer Dauer verkürzt, aber in informativere multidimensionale Signale konvertiert. So lassen sich effizient genaue Ergebnisse mit dem neu vorgeschlagenen Kernoperator, der auf der Verstärkungsmethode (engl. boosting method) basiert, gewinnen.

Abschließend werden die Kernmethoden zur Klassifikation biologischer Daten eingesetzt, die ein unbekanntes additives Rauschen aufweisen. Ein Beispiel solcher Signale sind Aktivitätsprofile im erzwungenen Schwimmtest (engl. Forced Swimming Test, FST) für die Klassifikation des Tierverhaltens. Auf beschränkten Impulsantworten (engl. Finite Impulse Response, FIR) basierend, werden in dieser Arbeit Klassifikatoren für das Tierverhalten implementiert, in denen das additive Rauschen durch einen Filter für beschränkte Impulsantworten (FIR-Filter) von den Aktivitätsprofilen entfernt wird. Die Parameter der FIR-Filter werden ermittelt, indem die Genauigkeit maximiert wird, mit der ein Klassifikator zwei Klassen des Aktivitätsprofiles zu separieren versucht. Die hier vorgestellte Methode zur Klassifikation von Verhaltensmustern führt zu einer zuverlässigen Unterscheidung verschiedener Klassen von Antidepressiva (Imipramine und Desipramine).

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor, Professor Dr. Andreas Zell, founding director of the Centre for Bioinformatics Tübingen (ZBIT). He gave me the opportunity as a research assistant and let me work in the field of Bioinformatics. His wide knowledge and his logical way of thinking have been of great value for me. His understanding, encouraging and personal guidance have provided a good basis for the present thesis. He patiently corrected my papers and dissertation and financially supported me during my work and let me attend at various conferences. I am deeply thankful to Prof. Dr. Oliver Kohlbacher for the time he spent to evaluate this dissertation.

In the technical work, Mr. Klaus Beyreuther gave me much help with the preparation of software and hardware, especially some special technical support. Thanks are also due to my other colleagues in the Department of Computer Architecture, such as Claudia Walter, Vita Serbakova, Dr. Hashem Tamimi, Andreas Dräger, my roommate Xiang Li, as well as those who I cannot all name here.

I owe my loving thanks to my wife Mitra Asadollahi, and also to my sons, Soroush and Shayan Mohammad Beigi. My wife cared for my children and patiently tolerated my long working hours and gave all the motivation and encouragement for helping me to focus on research and extended all her support during my final year of PhD life.

My special gratitude is due to my parents, brothers, my sister and their families for their loving support and weekly calls. Without their encouragement and understanding it would have been impossible for me to finish this work. Finally, I wish to express my warm and sincere thanks to Mr. Mojtaba Eslamian who accepted my responsibilities of the previous job in Fonoonteb company in Iran and continued his support during my PhD career.

Contents

Chapter 1

Introduction

1.1 Motivation

1.1.1 High-Dimensional Biological Data

The emergence of various new application domains, such as bioinformatics, underscores the need for analyzing high-dimensional data, specially when there are more variables (or features) than observations. In a gene expression microarray data set, there could be tens or hundreds of dimensions showing the expression level of exons or genes, each of which corresponds to an experimental condition, or in biosonar data classification, there are a limited number of echoes with thousands of data points to be classified.

In high-dimensional data analysis, one is often facing the problem that real data is noisy, and in many cases, features that are not informative for understanding the data structure itself or for performing later tasks, such as clustering, classification and regression. The combination of noise and very high dimensions (> 1000) presents challenges for data analysis and calls for efficient machine learning methods that take the inherent specifications and structure of natural data into account.

Mathematically, the learning problem can be described as finding a general rule or estimating a functional dependency that explains data given only a sample of limited size, and the learning process is a process of choosing an appropriate function from a given set of functions.

In this thesis, the main task for high-dimensional biological data is to find robust, fast and precise learning methods for noisy and incomplete data while considering the structure and dimension of the data.

The range of natural science topics addressed in this thesis is relatively broad and touches different areas from natural science such as genome biology, neuroscience and neurophysiology. We here present a brief overview over the main problem of each of them:

Imbalance Problem in Protein Data Classification Many classifiers are designed with

the assumption of well-balanced data sets. But in real problems, like protein classification and remote homology detection, when using binary classifiers like support vector machines (SVM) and kernel methods, we are facing imbalanced data in which we have a low number of protein sequences as positive data (minor class) compared with negative data (major class). With imbalanced data, the classifiers tend to classify almost all instances as negative. There have been two types of solutions for coping with imbalanced data sets. The first type, as exemplified by different forms of re-sampling techniques, tries to increase the number of minor class examples (oversampling) or decrease the number of major class examples (undersampling) in different ways. The second type adjusts the cost of error or decision thresholds in classification for imbalanced data and tries to control the sensitivity of the classifier [100, 69, 68, 131].

G-protein coupled receptors (GPCRs) are a large superfamily of integral membrane proteins that transduce signals across the cell membrane. Because of that important property and other physiological roles undertaken by the GPCR family, they have been an important target of therapeutic drugs. The function of many GPCRs is not known and accurate classification of GPCRs can help us to predict their function. In our study we want to classify GPCRs at the subfamily and sub-subfamily level. At this level in some sub-subfamilies, we have only a very low number of protein sequences as positive data (minor class) compared with others (major class). Then in this problem we are not only faced with high-dimensional protein sequences but also with the imbalance number of proteins in each class.

Biosonar Data Classification Bats can distinguish objects by emitting a series of ultrasound signals (chirps) that generally sweep covering frequencies from 22 to 100 kHz. To unravel the mechanism of echolocation, inspired by the bat biosonar system, researchers have utilized biosonar head and ultrasonic sensing techniques similar to that of bats for mobile robots (biomimetic robots) and tried to classify different textures and landmarks through their received echo signals [91, 82]. From our experiments and the work of other researchers [91, 82, 97, 48], we conclude that finding robust feature for classification is not trivial. For example, the orientation of trees as landmarks can result in large changes in the reflected echoes. Hence, in this case the only temporal based features can be inefficient, and the local temporal similarities between different echoes of one object as an indication of its texture is a significant issue that should be considered. Here, the reflected echo contains subsequence similarities, at random positions, representing the texture of an object (tree). Machine learning based approaches capable of extracting the sub-similarities are therefore an option to address this problem.

Animal Behavior classification in Forced Swimming Test The Forced Swimming Test (FST) is a behavioral test used frequently to evaluate the potential efficacy of drugs affecting the central nervous system (CNS) in rats or mice [104]. In this experiment, rats are exposed to a 15-min pretest swim period and followed the next day by a 5-min test swim. Immersion of rodents in water for an extended period of time produces a characteristic

behavior called immobility, in which the rat makes only those movements necessary to keep its head above the water. When antidepressant drugs are administered between the pretest and test periods, usually three times within 24 hours, the behavioral immobility is selectively decreased [24]. Depending on the type of drug, rats show a mixed behavior of activities such as immobility, struggling/climbing (the rat tries to escape from the water) and swimming. Researchers have tried to conclude the effect of drugs from the above three states (immobile, struggling and swimming). Typically, tricyclic antidepressants and drugs with selective effects on noradrenergic transmission increase struggling/climbing behavior, while selective serotonin reuptake inhibitors increase swimming behavior versus the control group [42, 36, 37]. In an automated classification method, we aim to classify animals treated with known antidepressants and the control group. However, our experiments show that the response of the rats to drugs is too complex to only consider those states (immobile, struggling and swimming) as indicator of the drug efficacy. Furthermore, those activity profiles (signals) inherently contain undesired and interference noise that should be removed before feature extraction and classification. Thereby, a learning method which allows the removal of the noise seems to be well suited.

1.1.2 Kernel Methods

The success of machine learning methods depends on their ability to solve pattern recognition and regression problems. The kernel methodology, founded on strong theoretical grounds, provides a powerful and unified framework for many disciplines. The intuitive idea behind kernel methods is that the data are nonlinearly mapped into a higher dimensional feature space, related to a nonlinear kernel function. In general, such a procedure has the advantage that data, which is linearly inseparable in the original space may become linearly separable in feature space. A kernel function represents a computational trick, which makes it possible to represent linear patterns efficiently in high–dimensional spaces, to ensure adequate representational power. It can also be considered as a measure of similarity; different kernels correspond to different notions of similarity. The structure of the data and our knowledge of the particular type of data suggest a way of comparison that we consider in our kernel function.

Any kernel methods solution comprises two parts: a module that performs the mapping into the feature space and a learning algorithm, designed to discover linear pattern in that space. The first class of methods that implemented the theory were Support Vector Machines (SVMs). They are general approximators (depending on the kernel) for the (nonlinear) relation that underlies the given sample pairs. Their successful performance have been drawing the attention of many researchers. Among other paradigms (like neural networks, Gaussian Processes) they became a worthy competitive with strong capabilities. They are conceptually simple, very transparent, and less sensitive to high input dimensionality.

1.2 Own Contributions

Since we believe that kernel methods offer a good possibility to tackle the above described problems, the main goal of this thesis is to find a practical way to extend and improve those methods for the problems at hand. As kernels methods created a new playground with room for extensions and improvements, this thesis is an attempt to contribute in that matter. Thereby, in summary we see our contributions to the individual topics as follows:

Imbalanced Problem in Protein Data To increase the accuracy of remote homology detection by discriminative methods, researchers also focused on finding new kernels, which measure the similarity between sequences, as main part of SVM based classifiers. So, after choosing an appropriate feature space, and representing each sequence as a vector in that space, one takes the inner product between these vector-space representations. For GPCRs classification, we use the local alignment kernel (LA kernel) that has been shown to have better performance compared with other previously suggested kernels for remote homology detection when applied to the standard SCOP test set [64, 109]. It represents a modification of the Smith-Waterman score to incorporate sub-optimal alignments by computing the sum (instead of the maximum) over all possible alignments.

In this work, we propose an oversampling technique for protein sequences. **S**ynthetic **P**rotein **S**equence **O**versampling (SPSO) involves creating synthetic protein sequences of the minor class, considering the distribution of that class and also of the major class, and it operates in data space instead of feature space. We show that in our method the information of the minor class increases. To show the efficiency of our methods, we use the G-protein coupled receptors (GPCRs) family and create artificial data based on it and then use our algorithm for both real and artificial data. Furthermore, we see how our algorithm can be used along with a different error cost method to increase the sensitivity and stability of the classifier.

Biosonar Data Classification We study the classification of biosonar signals as an example of random process signals which contain local similarities. Inspired by the solutions for remote homology detection in protein families and the string kernel proposed by Lodhi et al. [90], we suggest a similar kernel to measure the similarity of two time series. The p-length subsequence of that kernel simply measures the occurrences of fixed p-length subsequences for each of the time series in consideration. The more time series share similar p-length subsequences, the more similar they are. We also implement a more general kernel, which considers warping in the subsequences. It measures the whole similarities of all warped non-contiguous subsequences of the two time series, independent of their positions.

Kernel Selection in Time-series Kernels Having a set of the kernels for similarity extraction in time-series for different sizes of subsequences, we propose a method to find an optimal linear combination of the kernels. We find the optimal kernel selection via max-

imizing the Kernel Fisher Discriminant (KFD) criterion [92] to build the optimal linear combination of kernels. Given that criterion, to solve the optimization problem, we use a search method called Mesh Adaptive Direct Search (MADS).

Boosting method for Biosonar Data classification Gradient boosting is a machine learning approach, which builds one strong classifier from many base learners. Originally, boosting has been proposed in the 90's (Freund and Schapire, 1996 [45]) as a method for classification and regression, in which a fitting method or estimator, called the base learner, is fitted multiple times on re-weighted data and the final boosting estimator is then constructed via a linear combination of those base learners. We present two kinds of base learners for the gradient boosting: Ordinary Least Squares (OLS) and kernel-based base learners. Compared with our previous works, in which we presented a time resolved spectrum kernel to extract the similarities between echoes, we get more efficient and accurate results with the newly proposed boosting method. We compare the methods in terms of sensitivity, specificity, accuracy and Matthew's correlation coefficient and also the runtime of training and testing.

Animal Behavior classification in Forced Swimming Test In our work, we consider that the activity profiles (signals) inherently contain undesired and interference noise that should be removed before feature extraction and classification. We use a Finite Impulse Response (FIR) filter to filter out that additive noise from the activity profile. The parameters of the FIR filter are obtained via maximizing the accuracy of a classifier that tries to make a discrimination between two classes of the activity profiles (e.g. drug vs. control). We use the kernel Fisher discriminant criterion as a criterion for the discrimination, the DIviding RECTangles (DIRECT) search method for solving the optimization problem and Support Vector Machines (SVMs) for the classification task. We also consider the activity profiles as outputs of a black box and all-pole model and use system identification methods to find the parameters of that model, and show that Autoregressive (AR) coefficients are suitable features for the extraction of the dynamic behavior of rats and also the classification of activity profiles. Our proposed behavior classification method provides a reliable discrimination of different classes of antidepressant drugs (imipramine and desipramine) administered to rats versus a vehicle-treated group.

1.3 Organization of this Thesis

The organization of this thesis is as follows: To make this work more self contained, in the next chapter we begin with a short overview over the basic theoretical concepts that are necessary to understand the kernel-based algorithms employed in this work. We present some elements of statistical learning and regularization theory and introduce the concept of kernel functions. In chapter 3, we present kernel-based learning algorithms, including SVMs for classification, which are utilized in this thesis, from two perspectives:

regularization theory, and the more common geometric perspective.

Beginning from chapter 4 we present our own work: In chapter 4 the **S**ynthetic **P**rotein **S**equence **O**versampling (SPSO) method is explained. We evaluate the efficiency of our method for both artificial and real data.

In chapter 5 we present time-series kernels to measure the similarity of echoes as time series. We then use those kernels directly in a SVM-based classifier. The results show that those kernels allow for a very reliable discrimination of reflected sonar echoes from different objects.

In chapter 6 we propose a new method to find an optimal linear combination of the time-series kernels. We formulate the optimal kernel selection via maximizing the Kernel Fisher Discriminant criterion (KFD) and use the Mesh Adaptive Direct Search (MADS) method to solve the optimization problem.

In chapter 7 we study the efficiency of boosting methods for our classification task. We use the gradient boosting method with two kinds of base learners. The first one uses the Ordinary Least Squares (OLS) regression and the other one uses the kernel function as base learner.

In chapter 8 we propose FIR-based classifiers for animal behavior classification. Our proposed behavior classification method provides a reliable discrimination of different classes of antidepressant drugs (imipramine and desipramine) administered to rats versus a vehicle-treated group.

Finally, in chapter 9 we draw the conclusions from this work.

Chapter 2

Mathematics of Learning

2.1 Introduction

Mathematically, the learning problem can be described as finding a general rule or estimating a functional dependency that explains data given only a sample of limited size, and the learning process is a process of choosing an appropriate function from a given set of functions.

We learn from experience, this is commonplace. But we can be more specific: we learn by perceiving patterns and extrapolating them to other cases. In this sense, the learning process rests on generalization and works by generalizing perceived regularities.

Generalization, an important aspect of machine learning, is the ability of correctly classifying unseen data, which are not present in the training examples. Precisely, it is not sufficient for the algorithm to be consistent with only the training data, but also the algorithm must show the property of correctly classifying new examples. The case when a function becomes too complex in order to be consistent, is called overfitting. We should try to optimize the generalization and not the fitting on training data; in other words, there is a tradeoff between complexity and accuracy on training data and various methods have been proposed for choosing the optimal compromise [129, 130].

In supervised learning, we are given a sample of m training data, $x = (x_1, ..., x_m) \in \mathcal{X}$, together with a sample of corresponding outputs, $y = (y_1, ..., y_m) \in \mathcal{Y}$, and the collection of the labelled training sample, $\mathcal{D}_m = \{(x_i, y_i)\}_1^m$, used for training, and testing is a set of independently and identically distributed (*i.i.d.*) examples drawn by an unknown but fixed distribution ρ on $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Using the decomposition $\rho(x, y) = \rho(x)\rho(y|x)$, the sampling can be interpreted as a two steps process where first the input x according to $\rho(x)$ is sampled and then a corresponding output y is sampled with probability $\rho(y|x)$. While the first step can be totally random, the second step usually models the sampling of a noisy function f. So the relation between input and output spaces is probabilistic and not functional, and for a given input x, there is a distribution $\rho(y|x)$ on possible outputs. The goal is to learn a function $f \in \mathcal{H}$, $f : \mathcal{X} \mapsto \mathcal{Y}$ which models the probabilistic relation between $\mathcal{X}$ and $\mathcal{Y}$ in a way that $f(x) \approx y$.

The hypothesis space $\mathcal{H}$ represents the set of admissible functions the learning algorithm looks for; it is a subset of a larger space $\mathcal{T}$ called *target space*, which contains a broader class of functions from $\mathcal{X}$ to $\mathcal{Y}$; putting some constraints on the elements of $\mathcal{T}$ leads to the hypothesis space $\mathcal{H}$.

Common learning tasks are regression, where $\mathcal{Y} = \mathbb{R}$, and classification, in which $\mathcal{Y}$ is a set of c classes. For example, in binary classification $\mathcal{Y} = \{+1, -1\}$ and $c = 2$.

In this chapter, we briefly review the statistical learning theory, followed by an outline of the mathematical foundations of kernels. Precisely, the concepts of loss function, risk functional, Bayes function, regularization theory, Mercer kernel and reproducing kernel Hilbert space from Vapnik [129, 130], Hastie et al. [56] , Cucker and Smale [38], Poggio and Smale [103], are revised and some theorems such as Mercer's theorem and the Representer theorem are reported.

2.2 Loss Function and Expected Risk

A learning algorithm is a map from a data set $\mathcal{D}$ to a function f. We consider $f : \mathcal{X} \mapsto \mathcal{Y}$, which may be any function from $\mathcal{X}$ to $\mathcal{Y}$ [103].

Definition 2.1 (Loss function) *A loss function $l : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^+$ is a non negative function that measures the error $l(f(x), y)$ between the predicted output $f(x)$ and the actual output y.*

Common loss functions can be defined depending on the problem. For example, in regression problems the loss is usually a function of the difference between the target and the predicted value $l(f(x), y) = l(y - f(x))$. A typical example is the *quadratic*, or L_2 loss:

$$l(f(x), y) = (y - f(x))^2 \tag{2.1}$$

another example is the *absolute*, or L_1, loss:

$$l(f(x), y) = |f(x) - y| \tag{2.2}$$

and in the case of binary classification, with $y \in \mathcal{Y} = \{+1, -1\}$ the typical examples for the *misclassification loss* are *indicator loss*

$$l(f(x), y) = \theta(-yf(x)) = \begin{cases} 1 & \text{if } -yf(x) \leq 0 \\ 0 & \text{othewise} \end{cases} \tag{2.3}$$

and *Hinge loss*:

$$l(f(x), y) = (1 - f(x)y)_+ = \begin{cases} 1 & \text{if } -yf(x) \leq 0 \\ 1 - f(x)x & \text{othewise} \end{cases} \tag{2.4}$$

The aim of statistical learning theory [129, 130] is to define a risk functional, which measures the average amount of error of a hypothesis, and to look for a hypothesis among

the ones with lowest risk. If $l(f(x), y)$ is a loss function, measuring the error between the prediction $f(x)$ and the actual output y, then the average error is called the expected risk [103].

Definition 2.2 (Expected Risk) *Given a function $f \in \mathcal{T}$ and a loss function $l(f(x), y)$, the expected risk $err_\rho(f)$ of f with respect to distribution ρ is the expected loss*

$$err_\rho(f) = \int_{\mathcal{X}\times\mathcal{Y}} l(f(x), y)\rho(x, y)dxdy \tag{2.5}$$

Note that the expected error can almost never be precisely computed since we almost never know the distribution ρ. Nevertheless, we are looking for the minimizer f_ρ in some target space $\mathcal{T}$ such that:

$$f_\rho = \arg\min_{f\in T} \quad err_\rho(f) \tag{2.6}$$

This minimizer is called the *Bayes function* and its expected risk, called *Bayes risk*, is a lower bound on the error that depends only on the intrinsic difficulty of the problem. As the distribution ρ on $\mathcal{X} \times \mathcal{Y}$ is unknown and the expected risk cannot be explicitly computed, we approximate the expected risk by the *empirical error* (or sample error or empirical risk) on the data collection $\mathcal{D}$.

Definition 2.3 (Empirical Error) *Given a function $f \in \mathcal{T}$ and a loss function $l(f(x), y)$, the empirical error $err_{\mathcal{D}_m}(f)$ of f with respect to the data $\mathcal{D}_m$ is the average loss*

$$err_{\mathcal{D}_m}(f) = \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i) \tag{2.7}$$

The empirical error is a random variable depending on the random selection of the data $\mathcal{D}_m$. Since ρ is unknown, we can learn f by minimizing the empirical error (2.7). The essential question is whether the expected risk of the minimizer of the empirical error is close to the one of f_ρ.

2.3 Learning by Risk Minimization

Given a hypothesis space $\mathcal{H}$ and a training set $\mathcal{D}_m$, the Empirical Risk Minimization (ERM) is the method that finds the function

$$f_{\mathcal{D}_m} = \arg\min_{f\in\mathcal{H}} err_{\mathcal{D}_m}(f) \tag{2.8}$$

A nice property called *consistency*, which we would like to be valid, is that the expected risk of $f_{\mathcal{D}_m}$ tends to the expected risk of f_ρ, independently from the distribution ρ, when the number of training data tends to infinity:

$$\forall \rho \, \forall \varepsilon > 0 \, \lim_{m \to \infty} \Pr\{|err_\rho(f_{\mathcal{D}_m}) - err_\rho(f_\rho)| > \varepsilon\} = 0$$

When the consistency is valid, the ERM algorithms should find the best function in the hypothesis space. The following theorem guarantees the consistency over all f:

Theorem 2.1 (Vapnik and Chervonenkis, 1971) *ERM is consistent if and only if*

$$\forall \varepsilon > 0 \, \lim_{m \to \infty} \Pr\{\sup_{f \in \mathcal{T}} |err_\rho(f_{\mathcal{D}_m}) - err_\rho(f_\rho)| > \varepsilon\} = 0$$

Another set of desirable properties for ERM is that the mapping defined by ERM be *well–posed*. The mathematical term well-posed problem stems from a definition given by Hadamard [55]. He believed that mathematical models of physical phenomena should have the properties that:

Definition 2.4 (Well–posed Problem (Hadamard, 1902)) *A problem is well–posed if (1) a solution exists, (2) the solution is unique and (3) the solution depends continuously on the data. A problem is ill–posed if it is not well–posed.*

In general, the solution to ERM does not exhibit generalization, and because of the lack of uniqueness and stability, it is an *ill–posed* problem. It can be made well–posed by an appropriate choice of $\mathcal{H}$.

2.4 Regularization Theory

The regularization theory is a framework in which ill–posed problems can be solved by adding appropriate constraints on the solution. A general approach is to choose the hypothesis space $\mathcal{H}$ to be a convex set in a Hilbert space (see section 2.5):

$$\mathcal{H} = \{f : \Omega(f) \leq R^2\} \tag{2.9}$$

where $\Omega(f)$ is a convex function. For example, $\Omega(f) = ||f||^2$ where $||f||$ is the norm of f in the Hilbert space. So the well–posedness of the ERM problem can be recovered by adding constraints on the target space $\mathcal{T}$ to obtain the hypothesis space $\mathcal{H}$. There are two main approaches:

Ivanov regularization: The direct approach to restrict the hypothesis space such that the solution becomes unique and to find the solution of ERM consists in putting the constraint that f must be bounded [129, 130]:

$$\begin{array}{l} \min\limits_{f \in \mathcal{H}} \; \frac{1}{m} \sum\limits_{i=1}^{m} l(f(x_i), y_i) \\ \text{subject to} \quad ||f||^2 \leq R^2 \end{array} \tag{2.10}$$

Tikhonov regularization: The indirect approach adds a term estimating the complexity of the solution f to the empirical risk:

$$\min_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i) + \mu \phi(f) \tag{2.11}$$

The parameter $\mu > 0$ controls the tradeoff between the empirical error and the complexity of the function f. ϕ represents our prior knowledge about the function f. We consider $\phi(\cdot) = ||\cdot||_{\mathcal{H}} = ||f||^2$:

$$\min_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i) + \mu ||f||^2 \tag{2.12}$$

2.5 Mathematical Foundations of Kernels

In this section some useful concepts which represent the mathematical foundations of kernel machines will be introduced. We will characterize valid kernels and feature spaces, interpreting a kernel as the inner product in some feature space.

We follow two equivalent approaches: the first one uses Mercer's theorem to interpret the feature space as a Hilbert space of *real sequences*; the other one uses Reproducing Kernel Hilbert Spaces (RKHS) to interpret the feature space as a Hilbert space of functions. The section ends up with showing the general form of the solution of a Tikhonov regularized learning problem, which minimizes a cost functional composed by the error on training data and the complexity of the learned function.

2.5.1 Euclidean and Hilbert Spaces

At first, we define Euclidean (or inner product or pre–Hilbert) and Hilbert spaces, which represent an extension of Euclidean spaces from [38] and [103].

Definition 2.5 (Euclidean Space) *A Euclidean space $\mathcal{E}$ is a vector space with a bilinear map $\langle \cdot, \cdot \rangle : \mathcal{E} \times \mathcal{E} \mapsto \mathbb{R}$ such that $\forall f, g, h \in \mathcal{E}, a \in \mathbb{R}$*

1. $\langle f, g \rangle = \langle g, f \rangle$
2. $\langle f + g, h \rangle = \langle f, h \rangle + \langle g, h \rangle$ and $\langle af, g \rangle = a \langle f, g \rangle$
3. $\langle f, f \rangle \geq 0$ and if $\langle f, f \rangle = 0 \Leftrightarrow f = 0$

A Euclidean space is also a normed space with the norm induced by the inner product: $||f|| = \sqrt{\langle f, f \rangle}$.

Before we define the Hilbert space, we need the following definitions.

Definition 2.6 (Completeness) *A Euclidean space $\mathcal{E}$ is complete with respect to the norm induced by the inner product if*:
$\forall(f_1, f_2, ...) : f_n \in \mathcal{E}$ and $\lim_{n\to\infty} \sup ||f_n - f_m|| = 0 \Rightarrow \lim_{n\to\infty} f_n = f \in \mathcal{E}$

Definition 2.7 (Denseness) *A set A is dense in a set B if A intersects every nonempty open set in B.*

Definition 2.8 (Separability) *An inner product space is separable if it contains a countable dense subset.*

Finally, a Hilbert space is defined as a Euclidean space with some more properties:

Definition 2.9 (Hilbert Space) *A Hilbert space is an Euclidean space that is also (1) complete and (2) separable.*

Note that Hilbert spaces are generally infinite dimensional. We require the space to have a countable basis ϕ. We can write

$$f = \sum_{n=1}^{\infty} \alpha_n \varphi_n$$

for a basis $\{\varphi_n\}_{n=1}^{\infty}$.

As an example let $\mathcal{E}$ be the space of 2π–periodic functions with $\int_{-\pi}^{\pi} |f(x)|^2 dx < \infty$ and $\phi_n(x) = \exp(nxi)$ (the symbol i stands for the imaginary unit) then:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n \exp(nxi) \tag{2.13}$$

$$\langle f, g\rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)\overline{g(x)}dx \tag{2.14}$$

and

$$||f|| = \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} |f(x)|^2 dx} \tag{2.15}$$

A more general example of a Hilbert space is the set of square integrable functions on a compact set $\mathcal{X} \subseteq \mathbb{R}^d,\ d \in \mathbb{N}$

$$\mathcal{L}^2(x) = \left\{ f : \mathcal{X} \mapsto \mathbb{R} : \int_X f^2(x)dx < \infty \right\} \tag{2.16}$$

with the following inner product

$$\langle f, g\rangle = \int_{\mathcal{X}} f(x)g(x)dx \tag{2.17}$$

and also another example of a Hilbert space is the set of square convergent real sequences

$$l^2 = \left\{ (x_1, x_2, ...) : \sum_{i=1}^{\infty} x_i^2 < \infty \right\} \tag{2.18}$$

with the inner product

$$\langle x, z \rangle \doteq \sum_{i=1}^{\infty} x_i z_i \tag{2.19}$$

2.5.2 Mercer's Theorem

In this subsection, we first explain a valid kernel, then define an integral operator on valid kernels and also the feature space of that. At the end, we will see that the hypothesis space is a Hilbert space [103].

Definition 2.10 (Mercer Kernel): *A function* $K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ *is a Mercer kernel if*

1. K *is continuous*
2. K *is symmetric, i.e for all* $x, y \in \mathcal{X}$, $K(x, y) = K(y, x)$
3. K is *positive definite, i.e., for all finite set* $x_1, ..., x_m \subset \mathcal{X}$ *the* $m \times m$ *matrix with entries* $K(x_i, x_j)$ *is positive definite*

$$\forall m \in \mathbb{N}, \forall\ c_1, ..., c_m \in \mathbb{R} \sum_{i=1} \sum_{j=1} c_i c_j K(x_i, x_j) \geq 0$$

and a symmetric matrix is positive definite if all its eigenvalues are nonnegative

Definition 2.11 (Gram Matrix) *Given a Mercer kernel* K *and a set of objects* $\{x_1, ..., x_m\}$, *the* $m \times m$ *matrix* K *such that* $K_{ij} = K(x_i, x_j)$ *is called the Gram matrix of* K *with respect to* $\{x_1, ..., x_m\}$.

Theorem 2.2 (Integral Operator on Mercer Kernel) *The linear operator* $L_K : \mathcal{L}^2(\mathcal{X}) \mapsto \mathcal{L}^2(\mathcal{X})$ on a Mercer kernel K defined by

$$L_K f(x) \doteq \int_{\mathcal{X}} K(x, z) f(z) dz \tag{2.20}$$

is

1. *well–defined:* L_K *is continuous for all* f
2. *bounded:* $||L_K f|| \leq a||f||,\ a \in \mathbb{R}$
3. *positive definite:* $\int_{\mathcal{X}} \int_{\mathcal{X}} K(x, z) f(x) f(z) dx dz \geq 0$

The proof is based on the spectral theorem for compact linear operators on a Hilbert space (Cucker and Smale [38]).

Theorem 2.3 (Mercer's Theorem, 1909) *Given a Mercer kernel K on $\mathcal{X} \times \mathcal{X}$, let $\{\lambda_k,\ \varphi_k\}_{k=1}^{\infty}$ be a system of the eigenvalue/eigenfunctions of L_K with $\lambda_K \geq \lambda_{K+1} \geq 0$. Then for all $x, z \in \mathcal{X}$*

$$K(x,z) = \sum_{k=1}^{\infty} \lambda_k \varphi_k(x) \varphi_k(z) \tag{2.21}$$

where the convergence is uniform on $\mathcal{X} \times \mathcal{X}$ and absolute.

The following theorem from Cucker and Smale [38] shows what the feature map of a Mercer kernel is.

Theorem 2.4 (Feature Space of a Mercer Kernel) The feature map $\phi : \mathcal{X} \mapsto l^2$ defined as

$$\phi(x) = \{\sqrt{\lambda}\varphi_k(x)\}_{k=1}^{\infty} \tag{2.22}$$

is well–defined, continuous, and satisfies

$$K(x,z) = \sum_{k=1}^{\infty} \lambda_k \varphi_k(x) \varphi_k(z) = \langle \phi(x), \phi(z) \rangle \tag{2.23}$$

An important consequence is that a Mercer kernel can be interpreted as an inner product in the Hilbert space l^2 of real sequences. In addition, the Hilbert space l^2 of real sequences constitutes the *feature space* of our Mercer kernel. Note that if we are given a feature map $\phi(x)$ which we know to be in l^2 for all $x \in \mathcal{X}$, we can immediately build a valid kernel by setting $K(x,z) = \langle \phi(x), \phi(z) \rangle$ [38]. Now we define the set of square integrable functions on a compact set X associated with a Mercer kernel and show that it is a Hilbert space.

Definition 2.12 *Given a Mercer kernel K and its linear operator L_K defined in Equation* 2.20*, define*

$$H_K \doteq \left\{ f \in L^2(X) :\ f = \sum_{k=1}^{\infty} a_k \varphi_k \text{ with } \left(\frac{a_k}{\sqrt{\lambda_k}} \right) \in l^2 \right\} \tag{2.24}$$

where λ_k, φ_k are the eigenvalues and the eigenfunctions of L_K and define an inner product $\langle \cdot, \cdot \rangle_{H_K} : H_K \times H_K \mapsto \mathbb{R}$ as

$$\langle f, g \rangle_{H_K} \doteq \sum_{k=1}^{\infty} \frac{a_k b_k}{\lambda_k} \tag{2.25}$$

The elements of H_K are continuous functions and for all $f \in H_K$ the series

$$f = \sum_{k=1}^{\infty} a_k \varphi_k$$

converges uniformly and absolutely.

Theorem 2.5 (H_K is a Hilbert space)[103] *The hypothesis space H_K induced by a Mercer kernel K associated with its linear operator L_K defined in Equation* 2.20 *and with an inner product $\langle \cdot, \cdot \rangle_{H_K}$ defined in Equation* 2.25 *is a Hilbert space.*

Consequently given a Mercer kernel K, H_K is the Hilbert space generated by the eigenfunctions of the integral operator L_K. The set H_K of square integrable functions $\mathcal{L}^2(\mathcal{X})$ on a compact set $\mathcal{X}$ is the hypothesis space of a *kernel machine*, which is a learning algorithm that deals with the data only through Mercer kernels. The general form of the solution of a supervised learning algorithm with kernel machines is

$$f(x) = \sum_{i=1}^{m} a_i K(x_i, x) \tag{2.26}$$

for some coefficient $a_i \in \mathbb{R}$. By the definition of H_K, we mathematically characterized the hypothesis space of kernel machines.

2.5.3 Reproducing Kernel Hilbert Spaces

In the above, we characterized valid kernels and feature spaces, interpreting a kernel as an inner product in some feature space. It exploits Mercer's theorem to describe the feature space as a Hilbert space of real sequences. Another alternative approach uses the Reproducing Kernel Hilbert Spaces (RKHS) to interpret the feature space as a Hilbert space of functions. Both approaches are equivalent.

Definition 2.13 (Reproducing Kernel Hilbert Space) *A Reproducing Kernel Hilbert Space $\mathcal{H}_K$ is a Hilbert space of functions on a compact set $\mathcal{X}$. These functions have the properties that for each $x \in \mathcal{X}$ the evaluation functionals $\mathcal{F}_x$ defined as*

$$\mathcal{F}_x[f] = f(x), \ \forall \ f \in \mathcal{H} \tag{2.27}$$

are linear and bounded.
some properties of RKHS are:

- $F_x[f+g] = F_x[f] + F_x[g] = f(x) + g(x)$
- $\forall x, \ z \in \mathcal{X}, \ |K(x,z)| \leq \sqrt{K(x,x)}\sqrt{K(z,z)}$

- $\forall x \in \mathcal{X},\ |f(x)| \leq ||f||_{H_k} \sqrt{K(x,x)}$
- $\mathcal{H}_K$ is made of continuous functions
- $f = \sum_k a_k \varphi_k$; the series converges absolutely and uniformly in $\mathcal{X}$

Theorem 2.6 (Reproducing Property of a Hilbert Space) *Given a Mercer Kernel K, we define a function $K_x : \mathcal{X} \mapsto \mathbb{R}$ as*

$$K_x(z) = K(x,z) \tag{2.28}$$

Then we have:

- $K_x \in \mathcal{H}, \forall x \in \mathcal{X}$
- *for each RKHS there exists a unique Mercer kernel K called reproducing kernel*
- *conversely, for each Mercer kernel K there exist an unique RKHS that has K as its reproducing kernel [38]*

The reproducing property means:

$$\mathcal{F}_x[f] = \langle K_x, f \rangle_{\mathcal{H}_K} = f(x) \tag{2.29}$$

Now we show that H_K in 2.24 and $\mathcal{H}_K$ are the same. For that, we define an inner product in H_0 (2.24) and assume:

$$f = \sum_{i=1}^{\infty} \alpha_i K_{x_i} \quad \text{and} \quad g = \sum_{j=1}^{\infty} \beta_j K_{z_j}$$

then

$$\langle f, g \rangle_{H_K} = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \alpha_i \beta_j K(x_i, z_j) \tag{2.30}$$

Let $\mathcal{H}_K$ be the the completion of H_0 with the associated norm. Considering

$$\langle K_{x_i}, K_{z_j} \rangle_{\mathcal{H}_K} = K(x_i, z_j)$$

Then

$$\begin{aligned}\langle f, K_x\rangle_{\mathcal{H}_K} &= \langle\sum_{i=1}^{\infty}\alpha_i K_{x_i}, K_x\rangle_{\mathcal{H}_K} \\ &= \sum_{i=1}^{\infty}\alpha_i\langle K_{x_i}, K_x\rangle_{\mathcal{H}_K} \\ &= \sum_{i=1}^{\infty}\alpha_i K(x_i, x) \\ &= f(x)\end{aligned}$$

While before the data was mapped into series of real numbers, now they are mapped into functions which sit on x that generate the Hilbert space:

$$x \xrightarrow{\phi} \phi(x) = K_x$$

and each point is represented in the feature space by a function that measures its similarity with the other points.

On the other hand, from Mercer's theorem, since $f \in H_K$ (Eq. 2.24):

$$f(x) = \sum_{k=1}^{\infty} a_k\varphi_k(x)$$

It follows that:

$$\begin{aligned}\langle f, K_x\rangle_{H_K} &= \sum_{k=1}^{\infty} a_k\langle\varphi_k, K_x\rangle_{H_K} = \sum_{k=1}^{\infty}\frac{a_k}{\lambda_k}\int\varphi_k(z)K(x,z)dz \\ &= \sum_{k=1}^{\infty}\frac{a_k}{\lambda_k}(L_K\varphi_k)(x) = \sum_{k=1}^{\infty}\frac{a_k}{\lambda_k}\lambda_k\varphi_k(x) \\ &= f(x)\end{aligned}$$

Finally, after introducing the above results, we reach the following theorem from Cucker and Smale (2001) :

Theorem 2.7 (H_K **and** $\mathcal{H}_K$ **are equal)** *The Hilbert spaces* H_K *and* $\mathcal{H}_K$ *are the same space of functions on* X *with the same inner product:*

$$\mathcal{H}_K \equiv H_K \quad \text{and} \quad \langle\cdot,\cdot\rangle_{\mathcal{H}_K} \equiv \langle\cdot,\cdot\rangle_{H_K}$$

In summary, Mercer's theorem provides a concrete way to construct a RKHS. In essence, Mercer's theorem provides a coordinate basis of an RKHS.

2.5.4 Representer Theorem

Using the results derived in the previous sections, now it can be shown that the general solution of the Tikhonov regularized learning problem

$$\min_{f \in \mathcal{H}_K} \frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i) + \mu ||f||_K^2$$

is

$$f(x) = \sum_{i=1}^{m} a_i K(x_i, x)$$

where $\mu > 0$ is a parameter that controls the tradeoff between the empirical error and the complexity of the function f and the problem of finding an $f \in \mathcal{H}$ that minimizes the above regularized risk functional is turned into the problem of finding the best coefficients $a_i,\ i = 1, ..., m$.

Proof of Representer Theorem The proof is given in a simplified case when the loss function $l(f(x_i), y_i)$ is convex and also differentiable with respect to f (Kimeldorf and Wahba [80]). Define

$$H(f) = \frac{1}{m} \sum_{i=1}^{m} l(f(x_i, y_i) + \mu ||f||_K^2 \tag{2.31}$$

Since $f \in \mathcal{H}_K$, then

$$f = \sum_{k=1}^{\infty} b_k \varphi_k \quad \text{and} \quad ||f||_K^2 = \sum_{k=1}^{\infty} \frac{b_k^2}{\lambda_k} \tag{2.32}$$

We set the first derivative of $H(f)$ with respect to b_k to zero, then

$$0 = \frac{\partial H(f)}{\partial b_k} = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial l(f(x_i), y_i)}{\partial b_k} \varphi_k(x_i) + 2\mu \frac{b_k}{\lambda_k}$$

$$\Rightarrow b_k = \lambda_k \sum_{i=1}^{m} -\frac{1}{2\mu m} \frac{\partial l(f(x_i), y_i)}{\partial b_k} \varphi_k(x_i)$$

So

$$b_k = \lambda_k \sum_{i=1}^{m} a_i \varphi_k(x_i) \tag{2.33}$$

where

$$a_i = -\frac{1}{2\mu m}\frac{\partial l(f(x_i), y_i)}{\partial b_k} \tag{2.34}$$

if we put Equation 2.33 into Equation 2.32 and use Mercer's theorem:

$$\begin{aligned} f(x) &= \sum_{k=1}^{\infty} b_k \varphi_k(x) = \sum_{k=1}^{\infty} \lambda_k \sum_{i=1}^{m} a_i \varphi_k(x_i)\varphi_k(x) \\ &= \sum_{i=1}^{m} a_i \sum_{k=1}^{\infty} \lambda_k \varphi_k(x_i)\varphi_k(x) = \sum_{i=1}^{m} a_i K(x_i, x) \end{aligned}$$

In the case of a regression problem, if we use the quadratic loss $l(f(x), y) = (y - f(x))^2$, from Equation 2.34 we have:

$$a_i = -\frac{1}{2\mu m}\frac{\partial l(f(x_i), y_i)}{\partial b_k} = \frac{y_i - f(x_i)}{\mu m} = \frac{y_i - \sum_{j=1}^{m} a_j K(x_j, x_i)}{\mu m}$$

$$\Rightarrow y_i = \mu m a_i + \sum_{j=1}^{m} a_j K(x_j, x_i) \tag{2.35}$$

that in the matrix form becomes:

$$(\mu m \mathbf{I}_m + K)\mathbf{a} = \mathbf{y} \tag{2.36}$$

So, by means of the representer theorem, the problem of finding $f \in \mathcal{H}_K$ is turned into finding coefficients $a_i,\ i = 1, ..., m$. When $\mu \to 0$, $f(x) \to 0$; but setting μ not too small guarantees an unique solution of representer problem (2.31), because the matrix $(\mu m \mathbf{I}_m + \mathbf{K})$ has full rank, and a stable solution for the linear system (2.35) is well conditioned. In this thesis we use the representer theorem to bring the concept of the kernel function in a booting classifier.

2.5.5 Examples of Kernel Families

Until now we have seen that there are two methods to map the input data space:

1. Choose a map ϕ which *explicitly* gives us a Mercer Kernel k, or

2. Choose a Mercer kernel K which implicitly corresponds to a fixed mapping ϕ

Mathematically, kernels are often much easier to define and have the intuitive meaning of serving as a similarity measure between objects and structured data. Moreover, there exist simple rules for designing kernels on the basis of given kernel functions.

Basic Kernels

The most simple kernel between two vectors x, z is the inner product

$$K(x, z) = \langle x, z \rangle$$

More complex kernel may be constructed using simpler ones. The polynomial kernel is defined as

$$K_{pol}(x, z) = (\langle x, z \rangle + b)^d = \sum_{s=0}^{d} \begin{pmatrix} d \\ i \end{pmatrix} b^{d-s} \langle x, z \rangle^s$$

The feature space of $\langle x, z \rangle^s$ is indexed by all monomials i of degree s

$$\phi_i(x) = x_1^{i_1} x_2^{i_2} .. x_m^{i_m} \text{ subject to } \sum_{j=1}^{m} i_j = s$$

Another commonly used class of kernels is the class of Radial Basis Functions:

$$K_{RBF}(x, z) = \exp\left(-\frac{||x - z||}{2\sigma^2}\right)$$

in which the parameter σ controls the flexibility of the kernel in a similar way to the degree d in the polynomial kernel.

Kernels for structured data

A particular and interesting property of kernel functions is that they are closed under addition, multiplication with a positive constant and exponentiation [113, 117]. This allows the construction of new kernels from existing ones, which is especially interesting for the definition of kernels for structured objects. Some examples are as follows:

- Convolution kernels [57]: The basic idea behind convolution kernels is that the similarity between composite objects can be captured by a relation between the object and its parts. It defines the kernel function between input objects as the convolution of *sub-kernels*, i.e, the kernels for the decompositions of the objects. Convolution kernels are very general and can be applied in a various problems.

- Graph kernels: A graph is defined as a set of vertices and a set of edges. Graph kernels compare the structure of graphs such as the number of subgraphs they have in common. [86, 50, 49, 75].

- Diffusion kernels: The main idea behind diffusion kernels is that it is easier to describe the local neighborhood of an instance than describing the whole instance space [81, 132]. The neighborhood might be defined as instances that differs only by one property.

- Generative model kernels: The first prominent kernel based on generative models is the Fisher kernel [66, 67]. It is based on the gradient of log-likelihood of the generative model (a posteriori probability) with respect to the parameter. The Fisher kernel is defined from the information Fisher matrix deduced from the generative model. A general framework of defining generative-model kernels has been presented in [128]. The Fisher kernel comes as a particular case of the marginalized kernels.

- String kernels: The traditional kernel for text classification is simply the inner product of two words into the text space representation. A string kernel consists of comparing common subsequences in the two words. The gaps between the subsequences are penalized. This can be done using the total length in the two strings. The p-spectrum kernel [85, 109] counts how many contiguous sub-strings of length p the strings have in common. In this thesis we use this kernel and extent it for time series. We present time-series spectrum kernels to measure the similarity of biosonar signals as time series.

2.6 Summary

In this chapter, we presented the basic theoretical tool and mathematics of learning needed to understand the algorithms employed in this thesis. We explained the concepts of loss function, risk functional and regularization theory. We characterized valid kernels and features spaces, interpreting a kernel as the inner product in some feature spaces. We interpreted the feature space based on Mercer's theorem, as a Hilbert space of real sequences and based on Reproducing Kernel Hilbert Spaces (RKHS), as Hilbert spaces of functions and saw that both approaches are equivalent. We presented the general form of the solution of a Tikhonov regularized learning problem which minimizes a cost functional composed by the error on the training data and the complexity of the learnt function based on the representer theorem. The represent theorem guarantees that each function minimizing the regularized risk functional can be written down in a closed form as a linear combination of kernels evaluated at the training data only. In the next chapter we derive the SVM algorithm from two perspectives: Tikhonov regularization and the more common geometric perspective.

At the end we presented some important examples of kernel functions in vector spaces such as polynomial and RBF kernels and we saw that using a property of kernels functions that they are closed under addition, multiplication and exponentiation allows construction of new kernels from existing ones such as kernels for structured objects. Some examples were convolution kernels, Graph kernels, diffusion kernel and string kernels. Kernels can also be defined over more complex structures such as trees.

Chapter 3

Support Vector Machines and Kernel Methods

3.1 Introduction

Support Vector Machines (SVMs), originally developed by Vapnik and co–workers [25], are the most widespread kernel–based machine learning algorithms nowadays. They represent a very specific class of algorithms, characterized by the use of kernels, the absence of local minima, the sparseness of the solution and the capacity control obtained by acting on the margin or on other dimension independent quantities such as the number of support vectors. They provide a new approach to the problem of pattern recognition based on the statistical learning theory. SVMs are based directly on the results from the previous chapter and realize the principle of risk minimization. SVMs minimize the empirical risk simultaneously with a bound on the complexity, namely the margin. They always find a global optimum because of their formulation as a Quadratic Programming (QP) optimization problem with box constraints. Their simple geometric interpretation provides fertile ground for explaining how they work in a very easy manner.

SVMs are largely characterized by the choice of kernel which maps the inputs into a feature space in which they are separated by a linear hypothesis. Often the feature space is a very high dimensional one but the so–called curse of dimensionality problem is cleverly solved by turning to the statistical learning theory. The statistical learning theory tells us that learning in a very high dimensional feature space can be simpler if one uses a low complexity, i.e. simple class of decision rule [89]. All the variability and richness that one needs to have a powerful function class is then introduced by the mapping ϕ through the kernel function. In short, not the dimensionality but the complexity of the function matters. In addition, for certain feature spaces and corresponding mappings ϕ, there is a highly effective trick for computing scalar products in a high dimensional feature space using kernel functions. So SVMs represent a complete framework where several concepts are combined together to form a powerful theory: dimension independent generalization bounds, Mercer kernels and RKHS, regularization theory and QP optimization represent

therefore the foundations of SVMs.

SVMs are the best performing methods in various domains. They have been used frequently in different applications of systems biology and bioinformatics. For example, SVMs are used for prognosis or therapy outcome prediction based on microarray data. Golub et al. [51] and Mukherjee et al. [95, 96] applied SVMs to leukemia microarray data. Brown et al [27]. classified yeast genes into functional categories based on SVMs. Degroeve et al. [41] recognized the starts of introns by SVMs. Carter et al. [30] identified functional RNAs on genomic DNA by SVMs. In most Bioinformatics applications support vector machines improved previous results.

After their introduction in the mid 90s, the soft margin classifier was introduced by Cortes and Vapnik in 1995 [34] then the algorithm was extended to the regression case by Vapnik [129] and to clustering problems ([126], [112] and [20]). Two books written by Vapnik ([129], [130]) provide a very extensive theoretical background of the field. Other references can be found in Cristianini and Shawe-Taylor [35]; Shawe-Taylor and Cristianini [114]; Hastie et al. [56].

In this chapter, at first, we derive the SVM algorithm from two perspectives: Tikhonov regularization from the previous chapter, and the more common geometric perspective. Then, we will discuss the regression and clustering applications of the SVMs and also the optimization algorithms used in SVMs.

3.2 SVMs and Regularization theory

We start with Tikhonov regularization

$$\min_{f \in \mathcal{H}_K} \frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i) + \mu ||f||_K^2$$

where K and $\mathcal{H}_K$ are the Mercer kernel and hypothesis space respectively, and $l(f(x), y)$ is the hinge loss function [129]

$$l(f(x), y) = |1 - yf(x)|_+ \tag{3.1}$$

which is non-differentiable at $(1 - y_i f(x_i)) = 0$, and we cannot follow the approach in the proof of the representer theorem (chapter 2). But it is a convex function and so the representer theorem is still applicable. We introduce non negative slack variables ξ as follows

$$\xi_i = |1 - y_i f(x_i)|_+$$

and so

$$\xi_i \geq 1 - y_i f(x_i)$$

then the problem is converted to [114]:

$$\begin{aligned} \min_{f \in \mathcal{H}_K} \ & \frac{1}{m} \sum_{i=1}^{m} \xi_i + \mu ||f||_K^2 \\ \text{subject to} \quad & y_i f(x_i) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad i = 1, ..., m \end{aligned} \tag{3.2}$$

The SVMs contain an unregularized bias term b so the Representer theorem results in a function

$$f(x) = \sum_{j=1}^{m} c_j K(x_j, x) + b \tag{3.3}$$

By the Representer theorem we can rewrite the above constrained optimization problem as a constrained quadratic programming problem. Plugging $f(x)$ into the Equation 3.2 results in the *primal SVM* [35]:

$$\begin{aligned} \min & \ \frac{1}{m} \sum_{i=1}^{m} \xi_i + \mu \mathbf{c}^T K \mathbf{c} \\ \text{subject to} \quad & y_i (\sum_{j=1}^{m} a_j K(x_i, x_j) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad i = 1, ..., m \end{aligned} \tag{3.4}$$

Using Lagrange multiplier techniques, we derive the Wolfe dual quadratic program:

$$L(\mathbf{c}, \xi, b, \alpha, \zeta) = \frac{1}{m} \sum_{i=1}^{m} \xi_i + \mu \mathbf{c}^T K \mathbf{c} - \sum_{i=1}^{m} \alpha_i \left(y_i \left\{ \sum_{j=1}^{m} c_j K(x_i, x_j) + b \right\} - 1 + \xi_i \right) - \sum_{i=1}^{m} \xi_i \zeta_i \tag{3.5}$$

We want to minimize L with respect to c, b, ζ, ξ and α subject to the constraints of the primal problem and nonnegativity constraints on α and ζ_i. We first take partial derivatives with respect to b and ξ:

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \frac{1}{m} - \alpha_i - \zeta_i = 0 \Rightarrow 0 \leq \alpha_i \leq \frac{1}{m}$$

This results in a reduced Lagrangian:

$$L^R(\mathbf{c}, \alpha) = \mu \mathbf{c}^T K \mathbf{c} - \sum_{i=1}^{m} \alpha_i \left(y_i \sum_{j=1}^{m} c_j K(x_i, x_j) - 1 \right)$$

and after derivation with respect to $\mathbf{c}$:

$$\frac{\partial L^R}{\partial \mathbf{c}} = 0 \Rightarrow 2\mu K \mathbf{c} - KY\alpha = 0 \Rightarrow c_i = \frac{\alpha_i y_i}{2\mu}$$

where Y is a diagonal matrix whose i-th diagonal element is y_i; $Y\alpha$ is a vector whose i-th element is $\alpha_i y_i$. Substituting in the reduced Lagrangian for $\mathbf{c}$, we are left with the following *dual* program:

$$\begin{aligned} &\max_{\alpha\in\mathbb{R}^n} \sum_{i=1}^{m} \alpha_i - \tfrac{1}{4\mu}\alpha^T Q\alpha \\ &\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\qquad\qquad 0 \le \alpha_i \le \tfrac{1}{m} \quad i = 1, ..., m \end{aligned} \tag{3.6}$$

where Q is the matrix defined by

$$Q = \mathbf{y}K\mathbf{y}^T \Leftrightarrow Q_{ij} = y_i y_j K(x_i, x_j) \tag{3.7}$$

In most SVM applications, instead of the regularization parameter μ, regularization is controlled via a parameter C, defined using the relationship

$$C = \frac{1}{2\mu m}$$

Like μ, the parameter C also controls the trade-off between between fitting the data (empirical error) and the model complexity: a large value of C favors the empirical error, while a small value leads to a more regularized prediction function.

Using this definition and after multiplying our objective function by the constant $\frac{1}{2m}$ the basis regularization problem becomes

$$\min_{f\in\mathcal{H}_K} \; C\sum_{i=1}^{m} l(f(x_i), y_i) + \mu||f||_K^2 \tag{3.8}$$

and the primal and dual problems become, respectively:

$$\begin{aligned} &\min_{\mathbf{c}\in\mathbb{R}^n, \xi\in\mathbb{R}^n} \sum_{i=1}^{m} \xi_i + \tfrac{1}{c}\mathbf{c}^T Q\mathbf{c} \\ &\text{subject to :} \quad y_i(\sum_{j=1}^{m} a_j K(x_i, x_j) + b) \ge 1 - \xi_i \\ &\qquad\qquad \xi_i \ge 0 \quad i = 1, ..., m \end{aligned} \tag{3.9}$$

$$\begin{aligned} &\max_{\alpha\in R^n} \sum_{i=1}^{m} \alpha_i - \tfrac{1}{2}\alpha^T Q\alpha \\ &\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\qquad\qquad 0 \le \alpha_i \le C \quad i = 1, ..., m \end{aligned} \tag{3.10}$$

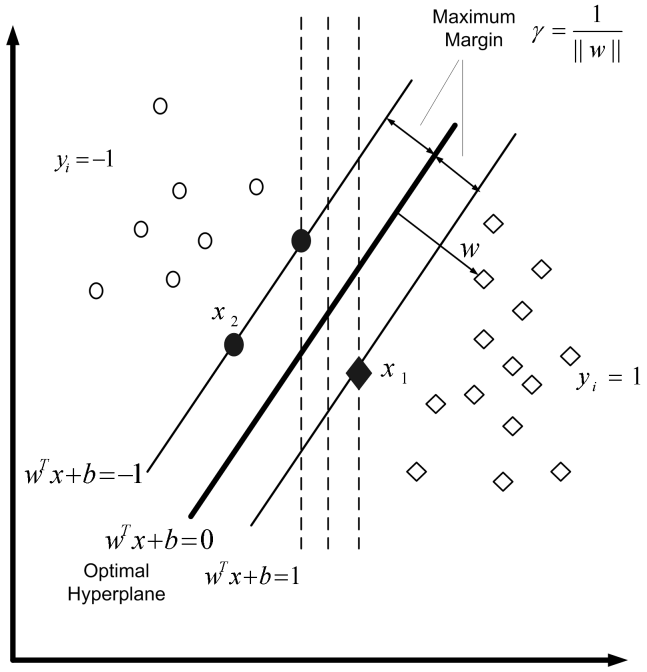


Figure 3.1: Optimal separating hyperplane in a two dimensional space. Two hyperplanes perfectly separate the data. However, the optimal one has a larger margin and intuitively would be expected to the more accurate on new observations.

3.3 SVMs from a geometric perspective

The traditional approach to developing the mathematics of SVMs is to start with the concept of separating hyperplane and margin. The theory is usually developed in a linear space, beginning with the idea of a perceptron that separates the positive and the negative examples. Defining the margin as the distance from the hyperplane to the nearest example, the basic observation is that intuitively, we expect a hyperplane with larger margin to generalize better than one with smaller margin (Fig. 3.1).

We denote our hyperplane by w and linear separating function as classification function by $sign(w^T x + b)$ (Fig. 3.1). It means that all positive examples are on one side of the boundary function $w^T x + b = 0$, and all negative examples are on the other side. At least one point exists for which $w^T x + b = 1$ and at least one point exists for which $w^T x + b = -1$. As the problem is linearly separable, there exists a weight vector w and b such that $y_i(w^T x_i + b) > 0$ $(i = 1, ..., n)$. Rescaling w and b such that the point(s) closest to the hyperplane satisfy $|w^T x_i + b| = 1$, we obtain a canonical form of the hyperplane satisfying $y_i(w^T x_i + b) \geq 1$. That means the margin in this case equals $\gamma = \frac{1}{||w||}$. This can be seen by considering two points x_1, x_2 on the opposite sides of the hyperplane which exactly satisfy $|w^T x_i + b| = 1$, projecting them onto the hyperplane normal vector $\frac{w}{||w||}$:

$$\begin{aligned} w^T x_1 + b &= 1 \\ w^T x_2 + b &= -1 \\ \Rightarrow w^T (x_1 - x_2) &= 2 \\ \Rightarrow \langle w/||w||, x_1 - x_2 \rangle &= \frac{2}{||w||} \end{aligned}$$

To maximize the margin γ, we have to maximize $\frac{1}{||w||}$ which means we have to minimize $||w||$ or equivalently $w^T w = ||w||^2$. To realize the risk minimization principle, we maximize the margin. This leads to the support optimization problem [118].

$$\begin{aligned} &\min_{w \in \mathbb{R}^n} \quad ||w||^2 \\ &\text{subject to} \quad y_i \left(\langle w, x_i \rangle + b \right) \quad i = 1, ..., n \end{aligned} \tag{3.11}$$

In addition, we need to work with data sets that are not linearly separable, so we introduce slack variables ξ_i, just as before. With the slack variables the primal SVM problem becomes

$$\begin{aligned} C \sum_{i=1}^{m} \xi_i + \tfrac{1}{2}||w||^2 \\ y_i \left(\langle w, x \rangle + b \right) \geqslant 1 - \xi_i \quad i = 1, ..., m \\ \xi_i \geqslant 0 \quad i = 1, ..., m \end{aligned} \tag{3.12}$$

Using Lagrange multipliers we can derive the same dual form as in the previous section (Equation 3.10).

3.3.1 Optimality conditions

The primal and dual are both feasible convex quadratic programs. They both have the same optimal and objective values. All optimal solutions obtained through the partial derivation with respect to parameters in Equation 3.5 must satisfy the Karush-Kuhn-Tucker (KKT) conditions [114]:

$$\begin{aligned}
\sum_{j=1}^{m} c_j K(x_i, x_j) - \sum_{j=1}^{m} y_i \alpha_j K(x_i, x_j) = 0 \qquad & i = 1, ..., m \\
\sum_{i=1}^{m} \alpha_i y_i = 0 & \\
C - \alpha_i - \xi_i = 0 \qquad & i = 1, ..., m \\
y_i \left(\sum_{j=1}^{m} y_j \alpha_j K(x_i, x_j) + b \right) - 1 + \xi_i \geqslant 0 \qquad & i = 1, ..., m \\
\alpha_i \left[y_i \left(\sum_{j=1}^{m} y_j \alpha_j K(x_i, x_j) + b \right) - 1 + \xi_i \right] = 0 \qquad & i = 1, ..., m \\
\zeta_i \xi_i = 0 \qquad & i = 1, ..., m \\
\zeta_i, \xi_i, \alpha_i \geqslant 0 \qquad & i = 1, ..., m
\end{aligned}$$

Typically the number of non zero α_i is much smaller than the number m of training examples, and so the Equation 3.3 can be computed efficiently by summing only on examples x_i for which $\alpha_i > 0$. These examples are called Support Vectors (SVs) and represent the critical elements of the training set; they summarize all the information contained in the data set.

The KKT complementary conditions permit to find the value of ξ_i for SVs. If $\alpha_i = C$, the corresponding x_i is called *bounded* SV and ξ_i has an arbitrary value, while for $0 < \alpha_i < C$ it is called *unbounded* SV and stays on the geometric margin, which measures the Euclidean distance of the points from the decision boundary in the input space. The KKT conditions also permit to compute the offset b

$$b = y_i - \sum_{j=1}^{m} y_i \alpha_i K(x_i, x_j)$$

So if we know the optimal α_i, we can determine b. For a better computational stability, we can take the average on unbounded SVs.

$$b = \frac{1}{|\{i : 0 < \alpha_i < C\}|} \sum_{i:0<\alpha_i<C} \left\{ y_i - \sum_{j=1}^{m} y_i \alpha_i K(x_i, x_j) \right\}$$

A proposition from Cristianini and Shawe–Taylor [35] shows the relation between the optimum solution of α_i and the geometric margin, which measures the Euclidean distance of the points from the decision boundary in the input space.

$$\gamma = \sqrt{\sum_{i:\alpha_i^*} \alpha_i^*}$$

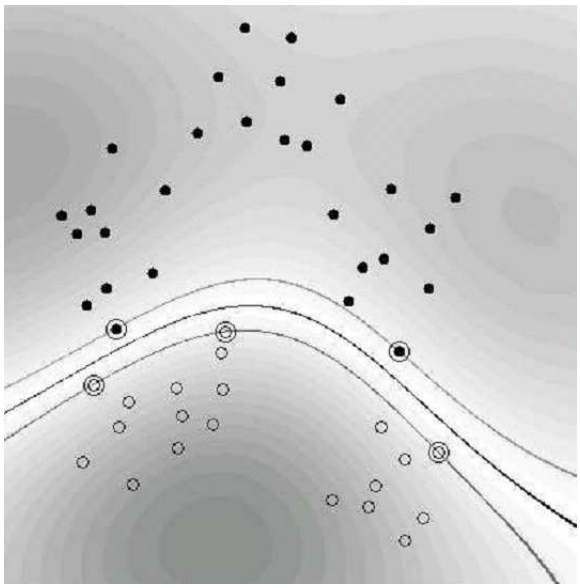


Figure 3.2: SVM classification with an RBF kernel. Support vectors are circled. From Schölkopf and Smola [113]

.

3.4 Support Vector Regression

Now we want to apply the support vector technique to regression. For the first time, in Vapnik [129], SVMs for regression problem with $\mathcal{Y} = \mathbb{R}$ were introduced. In his proposal, SVMs are the solution of the Tikhonov regularized problem but with a different loss function

$$l(f(x), y) = |y - f(x)|_\varepsilon = \max\{0,\ |y - f(x)| - \varepsilon\}$$

where $\varepsilon > 0$. In ε–SV regression the goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible [117]. Introducing slack variables ξ_i, ξ_i^* to penalize points that are above or below the ε–tube and considering the regression function $f = \langle w, \phi(x)\rangle + b$, the underlying optimization problem can be formulated as quadratic program:

$$\begin{array}{ll} \min\limits_{w,b,\xi,\xi^*} & \frac{1}{2}\langle w, w\rangle + C\sum\limits_{i=1}^{m}(\xi_i + \xi_i^*) \\ \text{subject to} & (\langle w, \phi(x_i)\rangle + b) - y_i \leqslant \varepsilon + \xi_i \\ & y_i - (\langle w, \phi(x_i)\rangle + b) \leqslant \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geqslant 0, \quad i = 1, ..., m \end{array} \tag{3.13}$$

where the constraints provide that the prediction will be close to the regression value

$$-\varepsilon - \xi_i^* \leqslant (\langle w, \phi(x_i)\rangle + b) - y_i \leqslant \varepsilon + \xi_i$$

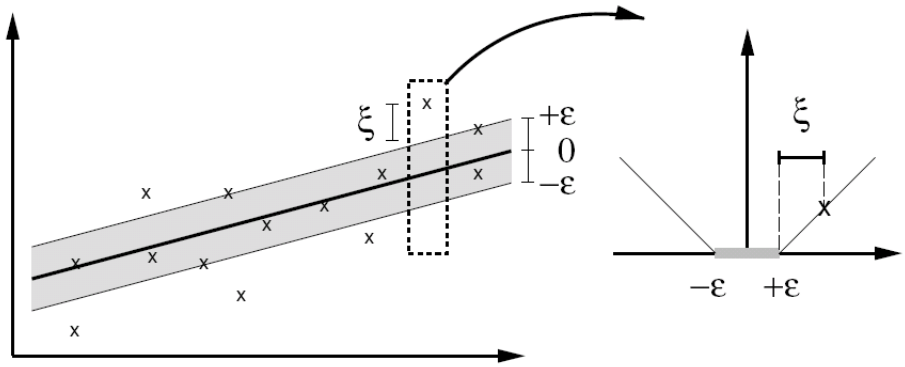


Figure 3.3: Support vector regression. A regression function is found such that a tube with radius ε around the regression function contains most of the data. Points outside the tube are penalized by ξ_i. Those penalties are traded off against the complexity given by $||w||$, which is represented by the slope of a linear function (From Schölkopf and Smola [113]).

In analogy to SVM classification, minimizing $||w^2||$ can also be understood as maximizing the so–called *ε–margin*, which is defined as the minimal distance between two patterns $\phi(x)$, $\phi^{'}(x)$ in feature space with $|f(x) - f(x^{'})| > 2\varepsilon$.

Similar to the SVM the above primal optimization problem(3.14), is a convex problem and applying the Kuhn–Tucker theory leads to its dual formulation

$$\begin{aligned} \min_{\alpha,\alpha^* \in R^m} \; & y^T(\alpha^* - \alpha) - \varepsilon e^T(\alpha^* + \alpha) - \tfrac{1}{2}(\alpha^* - \alpha)^T K(\alpha^* - \alpha) \\ \text{subject to} \quad & 0 \leqslant \alpha_i, \alpha_i^* \leqslant C, \quad i = 1, ..., m \\ & e^T(\alpha^* - \alpha) = 0 \end{aligned} \tag{3.14}$$

where $\alpha, \alpha^* \in \mathbb{R}^m$ are vectors of dual variables and e is the vector of all ones. The resulting weight vector can be written as

$$w = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*)\phi(x_i)$$

and the prediction function for regression is

$$f(x) = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*)K(x_i, x) + b$$

The Karush-Kuhn-Tucker (KKT) complementary conditions permit to find the values of ξ_i and ξ_i^* for SVs. The data points x_i with $\alpha_i = C$ or $\alpha_i^* = C$ are outside the ε–tube.

Data points with $0 < \alpha_i < C$ or $0 < \alpha_i^* < C$ are on the tube-border. Thus, the regression function only depends on the points lying outside or exactly on the ε–tube. The KKT conditions also permit to compute the value of b:

$$b = y_i - \langle w, \phi(x_i) \rangle + \varepsilon \tag{3.15}$$

Considering the generalization performance of the SVR, Shawe–Taylor and Cristianini [114] considered an upper bound on the expectation that the difference between the true function value y and the estimated one $\hat{y}$ at some point x exceeds a threshold γ:

Theorem 2.1 (Shawe–Taylor and Cristianini). *Let R be the smallest sphere enclosing the data in feature space. With probability at least $1 - \delta$ over the random draw of the data set $\mathcal{D}$, we have:*

$$E\left[\delta(|y - \hat{y}| > \gamma)\right] \leqslant \frac{1}{n(\gamma - \varepsilon)} \sum_i (\xi_i + \xi_i^*) + 4\sqrt{\frac{R^2 ||w||^2}{n(\gamma - \varepsilon)^2}} + 3\sqrt{\frac{\log(2/\delta)}{2n}}$$

3.5 Support Vector Clustering

Support Vector Clustering (SVC) also called one class SVMs, described in Tax and Duin [126] and [20], is an extension of basic SVMs. In this algorithm, one computes a set of contours which enclose the data points in terms of support vectors. These contours are interpreted as cluster boundaries. The outliers can be handled by relaxing the enclosing constraints and allowing some points to stay out of the contours. The primal problem of the SVC can be formulated as

$$\begin{gathered} \min_{R,c,\xi} \; R^2 + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad ||\phi(x) - c|| \leqslant R^2 + \xi_i \\ \xi_i \geqslant 0, \; i = 1, ..., m \end{gathered} \tag{3.16}$$

where R is the radius of the sphere, c is the center of the sphere, ξ_i are slack variables and C as regularization constant is a control parameter. The parameter C can be used to control the portion of examples separated by the sphere. Again, introducing Lagrange multipliers and applying the Kuhn-Tucker theory leads to the dual formulation

$$\begin{gathered} \max_{\alpha \in \mathbb{R}^m} \quad \sum_{i=1}^{m} \alpha_i K(x_i, x_i) - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j K(x_i, x_j) \\ \text{subject to} \quad 0 < \alpha_i < C, \quad i = 1, ..., m \end{gathered} \tag{3.17}$$

At each point, we define the distance of its image in feature space from the center of the sphere

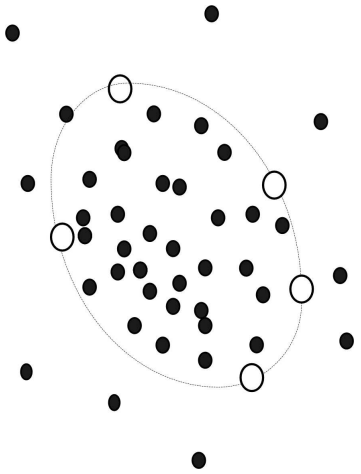


Figure 3.4: The sphere and SVs found by Equation 3.18 using a linear kernel on an example of data set.
.

$$R^2 = ||\phi(x) - c||^2$$

then, the KKT complementary conditions are satisfied by the optimal solution α^*, c^* and R^*:

$$\alpha_i^* \left[||\phi(x_i) - c||^2 - {R^*}^2 \right] = 0,\ i = 1, ..., m \tag{3.18}$$

This implies that only training examples x_i that lie on the surface of the optimal hypersphere have their corresponding α_i^* non–zero (support vectors) while for the remaining examples $\alpha_i^* = 0$. Furthermore, the distance of a point to the center of the sphere can be written as:

$$R^2(x) = K(x,x) - 2\sum_{i=1}^{m} \alpha_i K(x_i, x) + \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j K(x_i, x_j) \tag{3.19}$$

Therefore based on $R(x)$ and R^* one can decide whether x belongs to an inlier or an outlier of the data set.

3.6 Solving the SVM Optimization Problem

The SVM problem (3.9,3.10) is a Quadratic Programming (QP) optimization problem. The dual problem is easier to solve than the primal problem. We can solve the QP problem using standard software. However Q is a m by m dense matrix which can be hardly stored in memory for common problems with thousands of training data.

A lot of work has been done for finding efficient methods to solve the QP optimization for SVMs. Among them, decomposition strategies are commonly applied to solve the QP task associated with SVMs learning. The idea is to decompose the problem into a sequence of smaller problems. Then, the reduced problems can be solved by standard QP solvers. Examples of decomposition techniques and implementations have been discussed by Kaufmann [76], Joachims [72], Platt [101, 102], Osuna et al. [99], Keerthi et al. [77], Shevade et al. [115], Steinwart [119] and Bakir et al. [9].

The algorithm has an iterative nature and in each iteration, the variable $\alpha \in \mathbb{R}^m$ is split into a working space W and a fixed set F. Let $\mathcal{T}_W$ denote the indices of the working set and $\mathcal{T}_F$ be the indices of the fixed set such that $\mathcal{T}_W \cup \mathcal{T}_F = \{1, ..., m\}$ and $\mathcal{T}_W \cap \mathcal{T}_F = 0$. We can rewrite the dual problem (3.10) as:

$$\begin{aligned} \max_{\alpha_W \in \mathbb{R}^{|w|}, \alpha_F \in \mathbb{R}^{|F|}} & \sum_{\substack{i=1 \\ i \in W}}^{m} \alpha_i + \sum_{\substack{i=1 \\ i \in F}}^{m} \alpha_i - \tfrac{1}{2} \begin{bmatrix} \alpha_W & \alpha_F \end{bmatrix} \begin{bmatrix} Q_{WW} & Q_{WF} \\ Q_{FW} & Q_{FF} \end{bmatrix} \begin{bmatrix} \alpha_W \\ \alpha_F \end{bmatrix} \\ \text{subject to} \quad & \textstyle\sum_{i \in W} y_i \alpha_i + \sum_{i \in F} y_i \alpha_i = 0 \\ & 0 < \alpha_i < C, \quad i = 1, ..., m \end{aligned} \tag{3.20}$$

Then we treat α_W as variable and α_F as constant. Now we can solve the reduced dual problem:

$$\begin{aligned} \max_{\alpha_W \in \mathbb{R}^{|w|}} \quad & (1 - Q_{WF}\alpha_F)\alpha_W - \tfrac{1}{2}\alpha_W Q_{WW} \alpha_W \\ \text{subject to} \quad & \textstyle\sum_{i \in W} y_i \alpha_i = -\sum_{i \in F} y_i \alpha_i \\ & 0 < \alpha_i < C, \quad i = 1, ..., m \end{aligned} \tag{3.21}$$

The reduced problem is of fixed size and can be solve using a standard QP solver. An important issue in the decomposition algorithm is the way of selecting the working set. The basic idea is to examine points which are not in the working set and add the points which violate the reduced optimality to the working set. We remove points which are in the working set but far from violating the optimality conditions. Provided that the method for selecting of the working set satisfies those elementary conditions, the decomposition algorithm ultimately converges to the optimal solution. The convergence proof of various modifications of the decomposition algorithms can be found in [78, 88]. Two important decomposition algorithms include:

1-Decomposition algorithms with fixed size of working set, proposed by Osuna [99]. The working set size is fixed. In each iteration, a part of variables from the working set is replaced by previously fixed variables which violate the KKT conditions. This idea is used for instance in *SVMlight* by Joachims [72].

2-Sequential Minimal Optimizer (SMO) by Platt [101, 102]. The SMO is an extreme case of the general decomposition algorithm in which the working set contains just two

variables. The SMO approach has become very popular due to its implementational simplicity and a fast convergence. It is implemented for instance in the popular LIBSVM software package by Chang and Lin [31].

3.7 Summary

In this chapter, we introduced SVMs as a kernel–based machine learning algorithm based on the regularization theory which minimizes the empirical risk simultaneously with a bound on the complexity, namely the margin. We derived the SVM algorithm from two prospectives: Tikhonov regularization and a geometric perspective.

In the geometric representation, SVMs for classification rely on the idea of maximum margin hyperplane in feature space. The optimal hyperplane is uniquely obtained by QP optimization programming techniques. The solution is sparse and relies on examples called support vectors.

In support vector regression, the support vector technique is applied to regression and the Tikhonov regularization problem with a different loss function. It utilizes a special loss function, the ϵ–insensitive loss function. The regression function only depends on the points lying outside or exactly on the ϵ–tube.

In support vector clustering one can compute a set of contours which enclose the data points in terms of support vectors. Again, the data points that lies on the surface of the optimal hypersphere are called support vectors. This approach results in a state–of–the–art method for novelty detection.

At the end, we discussed methods for solving the SVM optimization. Decomposition strategies are commonly applied to solve the QP task; These decompose the problem into a sequence of smaller problems. We discussed some of the decomposition techniques, used in popular implementations of the SVMs.

In next chapters we will discuss the application of support vector machines and kernel methods for some biological problems. Beginning from the next chapter, we consider the imbalance data in a protein classification problem, when we use kernel methods. We will see how an oversampling technique can increase the accuracy of a kernel-based classifier.

Chapter 4

Kernel Methods for Imbalanced Protein data Classification

4.1 Introduction

4.1.1 Classification of G-protein Coupled receptors families

G-protein coupled receptors (GPCRs) are a large superfamily of integral membrane proteins that transduce signals across the cell membrane [4] (Fig. 4.1). Through their extracelluar and transmembrane domains they respond to a variety of ligands, including neurotransmitters, hormones and odorants. They are characterized by seven hydrophobic regions that pass through the cell membrane (transmembrane regions), as shown in Fig. 4.1. Each GPCR has an amino terminal (NH2 or N-terminal) region outside of the cell, followed by intracellular and extracellular loops, which connect the seven transmembrane regions, and also an intracellular carboxyl terminal (COOH- or C-terminal) region. GPCRs are involved in signal transmission from the outside to the interior of the cell through interaction with heterotrimeric G-proteins, or proteins that bind to guanine (G) nucleotides. The receptor is activated when a ligand that carries an environmental signal binds to a part of its cell surface component. A wide range of molecules is used as the ligands including peptide hormones, neurotransmitters, pancrine mediators, etc., and they can be in many forms: *e.g.*, ions, amino acids, lipid messengers and proteases [4, 60].

The function of many GPCRs are unknown and understanding the signaling pathways and their ligands in laboratory is expensive and time-consuming. But the sequence of thousands of GPCRs are known [21]. Hence, if we can develop an accurate predictor of the class (and so function) of GPCRs from their sequence it can be of great usefulness for biological and pharmacological research. According to the binding of GPCRs to different ligand types they are classified into different families. Based on GPCRDB (G protein coupled receptor data base) [21] all GPCRs have been divided into a hierarchy of ‘class’, ‘subfamily’, ‘sub-sub-family’ and ‘type’ (Fig. 4.2).

We want to classify GPCRs at the family, subfamily and sub-subfamily level. Because

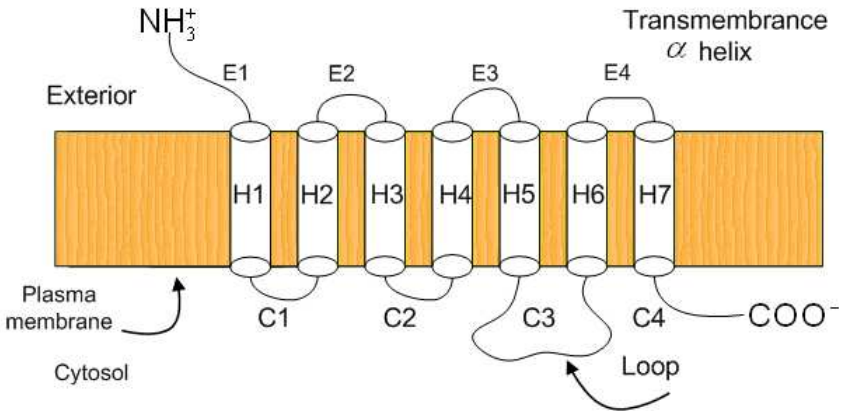


Figure 4.1: Schematic representation of GPCR shown as seven transmembrane helices depicted as cylinders along with cytoplasmic and extracellular hydrophilic loops.

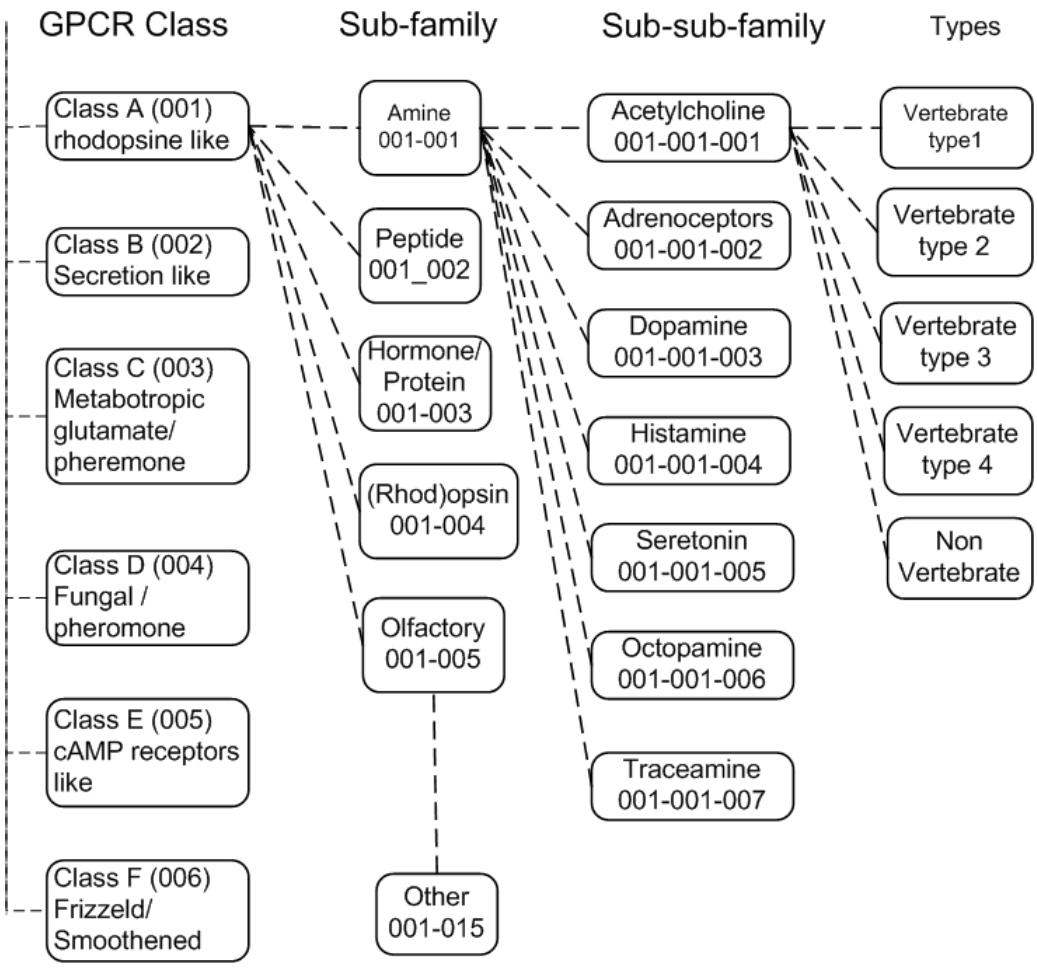


Figure 4.2: GPCR family tree according to GPCRDB nomenclature.

of the divergent nature of GPCRs it is difficult to predict the classification of GPCRs by means of sequence alignment approaches. The standard bioinformatics approach for function prediction of proteins is to use sequence comparison tools such as PSI-BLAST [3] that can identify homologous proteins based on the assumption of low evolutionary divergence, which is not true for GPCRs families. Here, we are facing a more difficult problem of remote homology detection, where classifiers must detect a remote relation between unknown sequence and training data.

There have been several recent developments to the classification problem specific to the GPCR superfamilies. Moriyama and Kim [79] developed a classification method based on discriminant function analysis using composition and physicochemical properties of amino acids. Elrod and Chou [43] suggested a covariant discriminant algorithm to predict GPCRs' type from amino acid composition. Qian et al. [106] suggested a phylogenetic tree based profile hidden Markov model (T-HMM) for GPCR classification. Karchin et al. [74] developed a system based on support vector machines built on profile HMMs. They generated fisher score vectors [65] as feature vectors for SVM classifier form those profile HMMs. They showed that classifiers like SVMs that are trained on both positive and negative examples can increase the accuracy of GPCRs classification compared with only HMMs as generative method.

To increase the accuracy of remote homology detection by discriminative methods, researchers also focused on finding new kernels, which measure the similarity between sequences as main part of SVM based classifiers. So after choosing an appropriate feature space and representing each sequence as a vector in that space, one takes the inner product between these vector-space representations.The Spectrum kernel [85], Mismatch kernel [84] and Local alignment kernel [64] are examples of those kernels and it has been shown that they have outperformed previous generative methods for remote homology detection.

Another important problem in classification of GPCRs is the number of proteins at the sub-subfamily level. At this level in some sub-subfamilies we have only a very low number of protein sequences as positive data (minor class) compared with others (major class). Some researchers have not considered those GPCRs families, or if they have included them in their classifier they did not get as good results for them as for other families with enough data [63].

4.1.2 Imbalanced dataset

Many classifiers are designed with the assumption of well-balanced datasets. But in real problems, like protein classification and remote homology detection, when using binary classifiers like support vector machines (SVMs) and kernel methods, we are facing imbalanced data in which we have a low number of protein sequences as positive data (minor class) compared with negative data (major class).

A dataset is imbalanced if the classes are not equally represented and the number of examples in one class (major class) greatly outnumbers the other class (minor class). With imbalanced data, the classifiers tend to classify almost all instances as negative. This

problem is of great importance since it appears in a large number of real domains, such as fraud detection, text classification, medical diagnosis and protein classification [84, 2]. There have been two types of solutions for coping with imbalanced datasets. The first type, as exemplified by different forms of re-sampling techniques, tries to increase the number of minor class examples (oversampling) or decrease the number of major class examples (undersampling) in different ways. The second type adjusts the cost of error or decision thresholds in classification for imbalanced data and tries to control the sensitivity of the classifier [100, 69, 68, 131]. Undersampling techniques involve loss of information but decrease the time of training. With oversampling we do not loose the information but instead it increases the size of the training set and so the training time for classifiers. Furthermore, inserting inappropriate data can lead to overfitting. Some researchers [2] concluded that undersampling can better solve the problem of imbalanced datasets. On the other hand, some other researchers are in favor of oversampling techniques. Wu and Chang [137] showed that with imbalanced datasets, the SVM classifiers learn a boundary that is too close to positive examples. Then if we add positive instances (oversampling), they can push the boundary towards the negative data, and we have increased the accuracy of the classifier.

To decide the question of oversampling vs. undersampling, two parameters should be taken into consideration: the *imbalance ratio* and the distribution of data in imbalanced datasets. The *imbalance ratio* ($\frac{NumberOfMinorityData}{NumberOfMajorityData}$) is an important parameter that shows the degree of imbalance. In undersampling we should be sure of the existence of enough information in the minor class and also of not loosing the valuable information in the major class. We found out that the oversampling technique can balance the class distribution and improve that situation. But the distribution of inserted positive instances is of great importance. Chawla et al. [32] developed a method for oversampling named Synthetic Minority Oversampling Technique (SMOTE). In their technique, between each positive instance and its nearest neighbors new synthetic positive instances were created and placed randomly between them. Their approach proved to be successful in different datasets.

On the other hand Veropoulos et al. [131] suggested using different error costs (DEC) for positive and negative classes. So the classifier is more sensitive to the positive instances and gets more feedback about the orientation of the class-separating hyperplane from positive instances than from negative instances.

In protein classification problems the efficiency of that approach (Veropoulos et al. [131]) has been accepted. In kernel based protein classification methods [85, 109, 84] a class-depending regularization parameter is added to the diagonal of the kernel matrix: $K'(x,x) = K(x,x) + \lambda n/N$, where n and N are the number of positive (or negative) instances and the whole dataset, respectively. But, based on our experiments, if the dataset is highly imbalanced and has overlapping data, choosing a suitable ratio of error costs for positive and negative examples is not always simple and sometimes the values near the optimum value of the error cost ratio give unsatisfying results.

4.1.3 Proposed Method for Imbalanced Protein Dataset

In this chapter, we show that a combination of the DEC method and our suggested oversampling method for protein sequences can increase the sensitivity and also stability of the classifier. We propose an oversampling technique for protein sequences in which the minority class in the data space is oversampled by creating synthetic examples. Working with protein data in data space instead of feature space allows us to consider the probability distribution of residues of the sequence using a HMM (Hidden Markov Model) profile of the minority class and also one of the majority class and then synthesize protein sequences which can push precisely the boundary towards the negative examples. So we increase the information of the minor class.

Synthetic **P**rotein **S**equence **O**versampling (SPSO) [12, 15] involves creating synthetic protein sequences of the minor class, considering the distribution of that class and also of the major class, and it operates in data space instead of feature space. Our method of oversampling (SPSO) can cause the classifier to build larger decision regions for the minor class without overlapping with the major class.

Kernel methods have widely been used for string classification. Examples of those kernels for text classification and remote homology detection in protein families include the spectrum kernel [86], mismatch kernel [84], and the string kernel proposed by Lodhi et al. [90]. For GPCR classification, we use the local alignment kernel (LA kernel)[11] that has been shown to have better performance compared with other previously suggested kernels for remote homology detection when applied to the standard SCOP test set [64, 109]. It represents a modification of the Smith-Waterman score to incorporate sub-optimal alignments by computing the sum (instead of the maximum) over all possible alignments. Using that kernel along with our oversampling technique we could get better accuracy and Matthew's correlation coefficient for the classification of GPCRs at the subfamily and sub-subfamily level than other previously published method.

In this work, we also create artificial data with different degrees of overlapping and imbalance ratio to show the efficiency of our methods. For that, we use the G-protein coupled receptors (GPCRs) family and create artificial data based on it. Furthermore, we see how our algorithm can be used along with DEC methods to increase the sensitivity and stability of the classifier.

In the following section, we explain the local alignment kernel. In section 4.3, we present the SPSO algorithm in details. In section 4.4, we explain the materials and dataset used in our study. The experimental results are given in section 4.5. Finally, we conclude in section 4.6.

4.2 Kernel function

In protein classification, variable length protein sequences must be converted to fixed length vectors to be accepted as input to a SVM classifier. These vectors should exploit prior knowledge of proteins belonging to one family and enable us to have maximum

discrimination for unrelated proteins. So the kernel function is of great importance for SVM classifiers in learning the dataset and also in exploiting prior knowledge of proteins and mapping data from input space to feature space. The Smith Waterman (SW) alignment score between two protein sequences tries to incorporate biological knowledge about protein evolution by aligning similar parts of two sequences but it lacks the positive definiteness as a valid kernel [109]. The local alignment kernel mimics the behavior of the Smith Waterman (SW) alignment score and tries to incorporate the biological knowledge about protein evolution into a string kernel function. But unlike the SW alignment, it has been proven that it is a valid string kernel. We used this kernel for our classification task, so we give a brief introduction to that algorithm:
If K_1 and K_2 are two string kernels then the convolution kernel $K_1 \star K_2$ is defined for any two strings x and y by:

$$K_1 \star K_2(x,y) = \sum_{x_1x_2=x, y_1y_2=y} K_1(x_1,y_2)K_2(x_2,y_2) \tag{4.1}$$

Based on work of Haussler [57] if K_1 and K_2 are valid string kernels, then $K_1 \star K_2$ is also a valid kernel. Vert et al. [64] used that point and defined a kernel to detect local alignments between strings by convolving simpler kernels.The local alignment kernel (LA) consists of three convolved string kernels. The first kernel models the null contribution of a substring before and after a local alignment in the score:

$$\forall (x,y) \in \chi^2, \;\; K_0(x,y) = 1 \tag{4.2}$$

The second string kernel is for alignment between two residues:

$$K_\alpha^{(\beta)}(x,y) = \begin{cases} 0 \quad \text{if } |x| \neq 1 \; or \; |y| \; \neq 1 \\ \exp[\beta s(x,y)] \quad \text{otherwise,} \end{cases} \tag{4.3}$$

where $\beta \geq 0$ controls the influence of suboptimal alignments in the kernel value and $s(x,y)$ is a symmetric similarity score or substitution matrix, e.g. BLOSUM62.
The third string kernel models affine penalty gaps:

$$K_g^{(\beta)}(x,y) = \exp\left\{\beta\left[g\left(|x|\right) + g\left(|y|\right)\right]\right\} \tag{4.4}$$

$g(n)$ is the cost of a gap of length n given by:

$$\begin{cases} g\left(0\right) = 0 & \text{if } n = 0, \\ g\left(n\right) = d + e\left(n-1\right) & \text{if } n \geq 1, \end{cases} \tag{4.5}$$

where d and e are gap opening and extension costs. After that the string kernel based on local alignment of exactly n residues is defined as:

$$K_n^{(\beta)}\left(x,y\right) = K_0 * \left(K_\alpha^{(\beta)} * K_\alpha^{(\beta)}\right)^{(n-1)} * K_\alpha^{(\beta)} * K_0. \tag{4.6}$$

This kernel quantifies the similarity of two strings x and y based on local alignments of exactly n residues. In order to compare two sequences through all possible local alignments, it is necessary to take into account alignments with different numbers n of aligned residues:

$$K_{LA}^{(\beta)} = \sum_{i=0}^{\infty} K_{(i)}^{(\beta)}. \tag{4.7}$$

The implementation of the above kernel can be done via dynamic programming [64].

4.3 SPSO: Synthetic Protein Sequence Oversampling

Given a set of positive training sequences (minor class) S_+ and a set of negative training sequences (major class) S_- we want to create synthetic protein sequences $S_{synthetic}$ as mutated replicas of each sequence of the minor class, provided that those synthetic sequences are created by an HMM profile (Hidden Markov Model profile) of the minor class and are phylogenetically related to that class and far away from the major class. For this, at first we build a multiple alignment of the sequences of the minor class using ClustalW [127] and then we train a hidden Markov model profile with length of the created multiple alignment sequences for each class (positive data and every family belonging to the negative data). For every sequence in the minor class we create another mutated sequence synthetically. For that, we consider an arbitrary N_m as number of start points for mutation in that sequence. We suppose the $HMMp_+$ (hidden Markov model profile of positive instances) has emitted another sequence identical to the main sequence until the first point of mutation. From that point afterward we assume that $HMMp_+$ emits new residues until the emitted residue is equal to a residue in the same position in the main sequence. From this residue, all residues are the same as residues in the original sequence until the next point of mutation (Fig. 4.3).

In this way, if the point of mutation belongs to a low entropy area of the HMM profile the emitted residue will be very similar to the main sequence (will have few mutations). We expect the emmitance probability of the synthesized sequence with $HMMp_+$ to be higher than with $HMMp_-$, if not (very rarely), we synthesize another one or we decrease the value of N_m. The N_m parameter can adjust the radius of the neighborhood of the original sequences and the synthesized sequences. With larger values of N_m, the algorithm creates sequences that are phylogenetically farer away from main sequences and vice versa. We used another routine to find a suitable value of N_m. At first, in the minor class, we find the protein sequence which has the highest emission probability with the HMM profile of the minor class and consider it as root node. Then, we suppose the root node has been mutated to synthesize all other sequences in the minor class through the $newSequence$ procedure of our algorithm. It means each sequence is a mutated replica of the root node sequence which is emitted by the HMM profile of the minor class. We

Algorithm SPSO(S_+,S_-)

Input : S_+, set of sequences of minority class; S_-, set of sequences of majority class

Output: $S_{synthetic}$, set of synthetic protein sequences from the minority class

Create HMM profile of set S_+, call it $HMMp_+$;
Create array of HMM profiles consisting of all families belonging to S_-, call it $HMMp_-[]$;
Choose an arbitrary number as number of start points for mutation, call it N_m;
for $i \leftarrow 1$ **to** $|S_+|$ **do**
    $s = S_+[i]$;
    repeat
        Create an array of sorted non-repeating random numbers with size N_m as array of start points for mutation, call it P_m ;
        $S_{synthetic}$[i]=`newSeq` (s,$HMMp_+$,P_m);
        $p_+ = P_e(S_{synthetic}[i], HMMp_+)$; `/* emittance probability of synthesized sequence by` $HMMp_+$ `*/`
        $p_-[] = P_e(S_{synthetic}[i], HMMp_-[])$;
    until $p_+ < \max p_-[]$;
end
return $S_{synthetic}$

Function `newSeq(`s,$HMMp_+$,P_m`)`

Input : s, original sequence; $HMMp_+$, HMM profile of set S_+ to which s belongs; P_m, array of start points for mutation

Output: $s_{synthetic}$, synthetic sequence from s

$s_{synthetic} = s$;
for $i \leftarrow 1$ **to** $|P_m|$ **do**
    $p = P_m[i]$; `/* assume that` $HMMp_+$ `in position` p `has emitted` $s[p]$ `*/`
    repeat
        $s_{synthetic}[p+1]$= emitted residue in position $p+1$ by $HMMp_+$;
        $p = p + 1$;
    until $(newres \neq s[p])$ && $(p < |HMMp_+|)$;
end
return $s_{synthetic}$

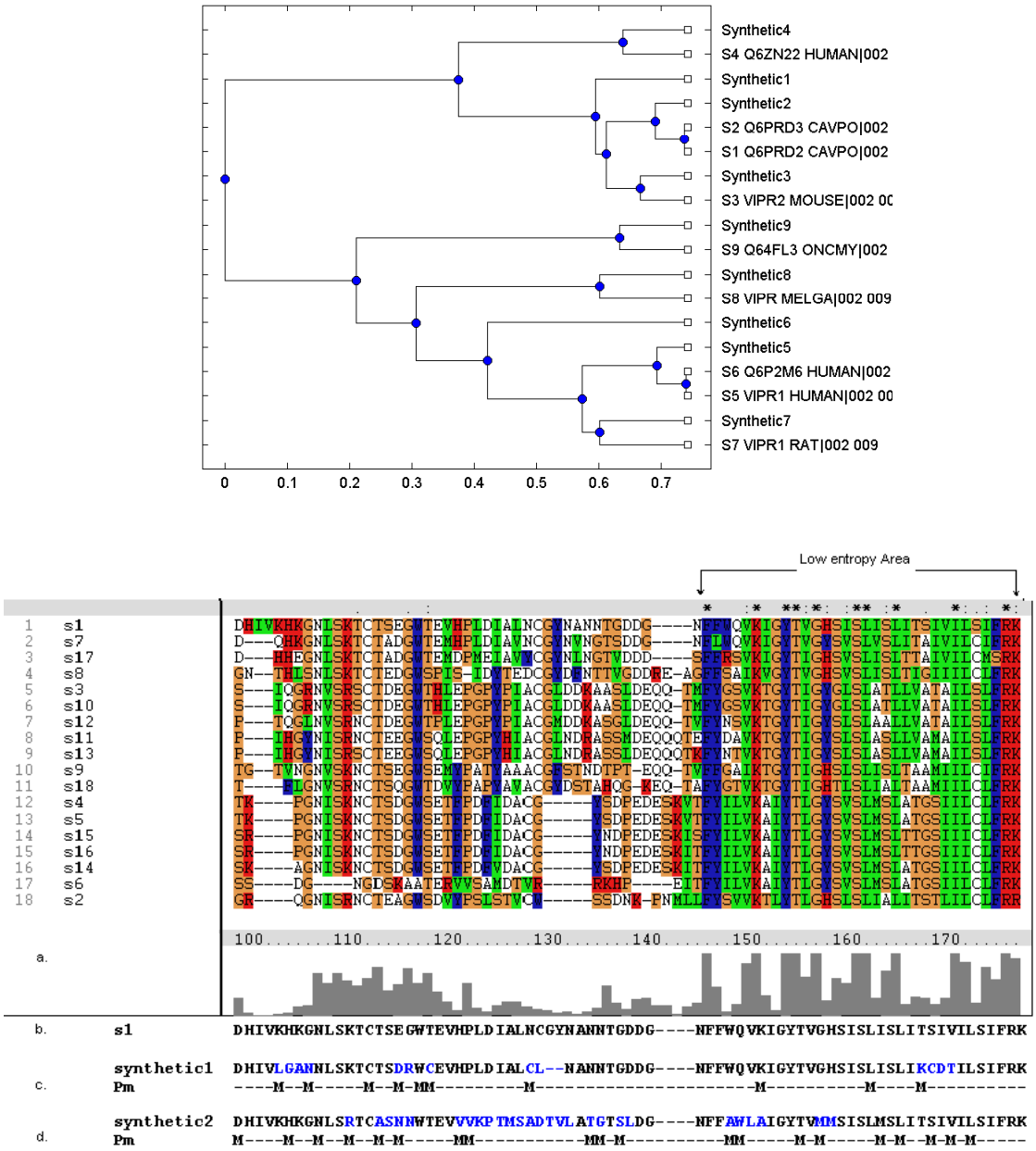


Figure 4.3: The phylogenetic tree of the original and the synthesized sequences from the "vasoactive intestinal polypeptide" family of GPCRs (**upper**) and an example of the SPSO algorithm for sequences from the above family (**lower**). **a.** Multiple sequence alignment and low entropy area of that family **b.** A part of sequence $s1$. **c.** Synthetic sequence of $s1$ with N_m=50. **d.** Synthetic sequence of $s1$ with N_m=100 (P_m: array of start points, shown by M, for mutations).

gain the value of N_m for each sequence. Then, we get the average of all those values as N_m entry for the SPSO algorithm.

With each call of the SPSO algorithm, we double the minor class. As an example of random synthesizing of sequences, Fig. 4.3(upper) shows the phylogenetic tree of the original sequences and the synthesized sequences for the vasoactive intestinal polypeptide family of class B (9 out of 18 sequences were randomly selected). It is shown that the synthesized sequences of most original sequences have less distance to them than to other sequences. In that figure (lower) we see two synthetic sequences of $s1$ with different values of N_m. In the low entropy area of the HMM profile of that family we have less mutations.

4.4 Datasets

To evaluate the performance of our algorithm, we ran our experiments on a series of both real and artificial datasets, whose specification covers different complexity and allows us to fully interpret the results. We want to check its efficiency with different ratio of imbalance and complexity. Fig. 4.4 shows the pictorial representation of our datasets. In the first one, the distribution of the positive and negative data are completely different and they are separate from each other. With that distribution, we want to see, how the imbalance ratio affects the performance of the classifier by itself. The second one shows datasets in which positive data are closer to negative data and there is an overlap between the minor and major classes. With this distribution, we can consider both the ratio of imbalance and overlap of the datasets in our study. The third one is a case where the minor class completely overlaps with the major class and we have fully overlapping data.

We used the G-protein coupled receptors (GPCRs) family as real data and then created artificial data based on it.

The dataset of this study was collected from GPCRDB and we used the dataset June 2005 release [21]. The six main families are: Class A (Rhodopsin like), Class B (Secretin like), Class C (Metabotropic glutamate/pheromone), Class D (Fungal pheromone), Class E (cAMP receptors) and Frizzled/Smoothened family. The sequences of proteins in GPCRDB were taken from SWISS-PROT and TrEMBL [8]. All six families of GPCRs (5300 protein sequences) are classified in 43 subfamilies and 99 sub-subfamilies.

If we want to classify GPCRs at the sub-subfamily level, mostly we have only a very low number of protein sequences as positive data (minor class) compared with others (major class). We chose different protein families from that level to cover all states of complexity and imbalance ratio discussed above (Fig. 4.4). In some experiments we made artificial data using those families and synthesized sequences from them (discussed later). We used numbers to show the level of family, subfamily and sub-subfamily. For example 001-001-002 means the sub-subfamily Adrenoceptors that belongs to subfamily of Amine (001-001) and class A (001).

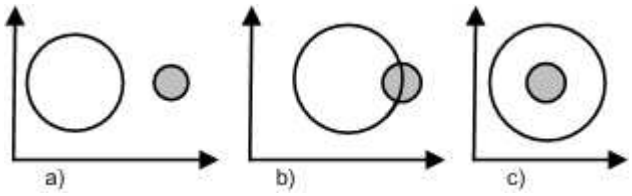


Figure 4.4: Pictorial representation of the minor (shaded circle) and major classes of our datasets.

4.5 Experiments

4.5.1 Artificial Data

We selected the peptide subfamily (001-002) of Class A (Rhodopsin-like) to classify its 32 families (or sub-subfamily level of class A). We built HMM profiles of all families and measured the probability of emission of sequences belonging to each one by all HMM profiles. We saw that the emission probability of each sequence generated by the HMM profile of its own family is higher than that of almost all other families. So we can conclude that the distribution of the peptide subfamily in a suitable feature map can be considered as in Fig. 4.4.a. We built a kernel matrix K for the training data. Each cell of the matrix is a local alignment kernel score between protein i and protein j (Fig. 4.5). Then we normalize the kernel matrix via $K_{ij} \leftarrow K_{ij}/\sqrt{K_{ii}K_{jj}}$. Each family is considered as positive training data and all others as negative training data. After that the SVM algorithm with RBF kernel is used for training. For testing, we created feature vectors by calculating a local alignment kernel between the test sequence and all training data. The number of sequences in the peptide subfamily is in the range of 4 to 251, belonging to (001-002-024) and (001-002-008), respectively. Thus the *imbalance ratio* varies from $\frac{4}{4737}$ to $\frac{251}{4737}$. Fig. 4.6.**a** shows the result of SPSO oversampling for classification of some of those families. We see that this method can increase the accuracy and sensitivity of the classifier faced with highly imbalanced data without decreasing its specificity. The minority class was oversampled at 100%, 200%, 300%,..., 800% of its original size. We see that the more we increase the synthetic data (oversample) the better result we get, until we get the optimum value. It should be noted that after oversampling, the accuracy of classifiers for the major class didn't decrease.

We compared our method with two other methods. The first one was SMOTE (Synthetic Minority Oversampling Techniques) [32] that operates in the feature space rather than in data space, so it works with all kind of data. The second comparison was done with randomly oversampling, in which we create random sequences by the HMM profile of each family. For this, like our method, we build a multiple alignment of the minor class sequences using ClustalW and then train a hidden Markov model profile with length of the created multiple alignment sequence. Then, we create random sequences by the HMM profile of each family. In this method we don't have enough control over the distribution

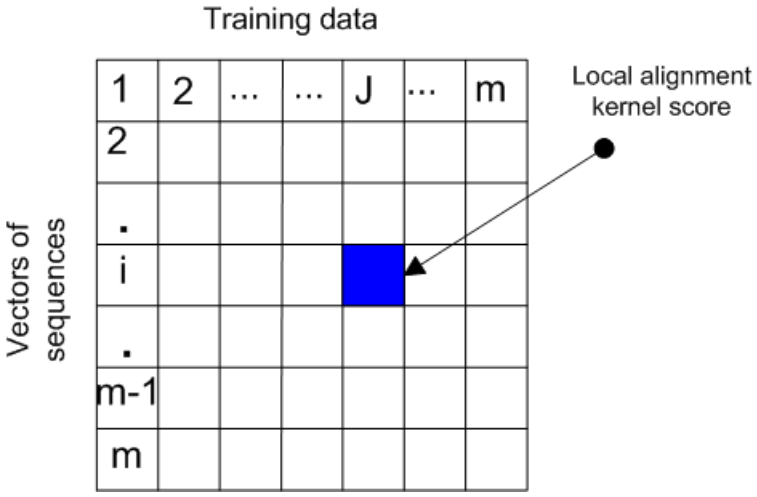


Figure 4.5: Calculating the kernel matrix of the training data.

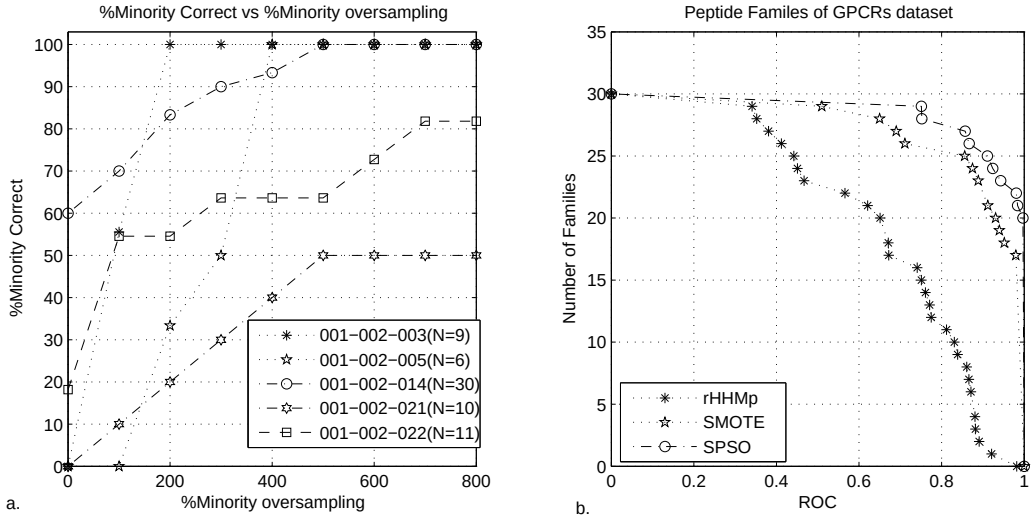


Figure 4.6: **a**. %Minority correct for SPSO oversampling for some families of peptide subfamily (N number of sequences). **b**. Comparison of several methods for oversampling. The graph plots the total number of families for which a given method exceeds an ROC score threshold.

of created random sequences. We call this method rHMMp.

In our study, we use the Bioinformatics Toolbox of MATLAB to create the HMM profiles of families and the SVMlight package [71], to perform SVM training and classification.

We used the Receiver Operating Characteristic (ROC) graphs [105] to show the quality of the SPSO oversampling technique. An ROC graph characterizes the performance of a binary classifier across all possible trade-offs between the classifier sensitivity (TP_{rate}) and false positive error rates (FP_{rate}) [124]. The closer the ROC score is to 1, the better performance the classifier has. We oversampled each minority class with the three different methods noted above, until we got the optimum performance for one of them. At that point, we calculated the ROC score of all methods.

Fig. 4.6.**b** shows the quality of classifiers when using different oversampling methods. This graph plots the total number of families for which a given method exceeds an ROC score threshold. The curve of our method is above the curve of other methods and shows better performance. In our method and in SMOTE, the inserted positive examples have been created more accurately than random oversampling (rHMMp). Our method (SPSO) outperforms the other two methods especially for families in which we have a low number of sequences, although the quality of SMOTE is comparable to the SPSO method.

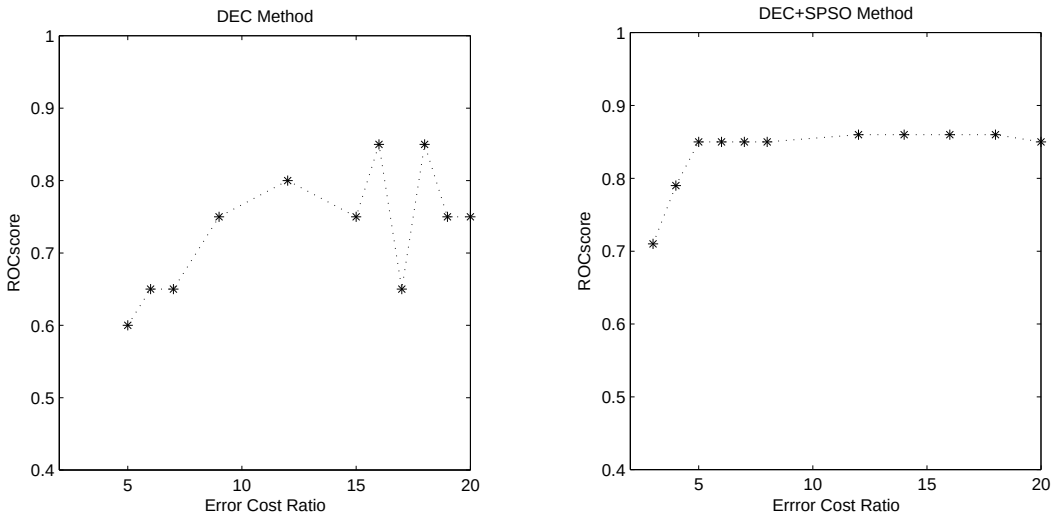
To study the second and third representation of the dataset shown in Fig. 4.4 we need to create some sequences synthetically. At first, we built the HMM profile of each family of the peptide families and then computed the probability score of each sequence when emitted not only by the HMM profile of its own family but also from all other families. The average of those scores for sequences of each family when emitted by each HMM profile can be used as a criterion for the closeness of the distribution of that family to other families and how much it can be represented by their HMM profiles. In this way we can find the nearest families to each peptide family. After that we synthesized sequences for each family through the $newSeq$ procedure of the SPSO algorithm, provided that it is emitted by the HMM profile of another near family and not by its own HMM profile. So after each start position for mutation (Fig.4.3 (lower)) we have residues that are emitted by another HMM profile (we want to have overlap with) instead of its own HMM profile and there is an overlap for the distribution of synthesized sequences between those two families. The degree of overlapping can be tuned by the value of N_m (number of mutations). This dataset (original and new synthesized sequences) can be considered as partially overlapping dataset (Fig. 4.4**.b**). If we create more sequences using other HMM profiles the distribution of the dataset is fully overlapping (Fig. 4.4**.c**). To study the partially overlapping datasets, we selected 10 families of peptide families and built the synthesized sequences as noted above. To create the fully overlapping dataset, we performed that routine for each family using the HMM profile of three families near to the original family, separately.

We compare our oversampling technique with the SMOTE oversampling technique and the different error cost (DEC) method [131]. Tables 4.1 and 4.2 show the results. We see that in general SPSO outperforms the SMOTE and DEC methods, and the perfor-

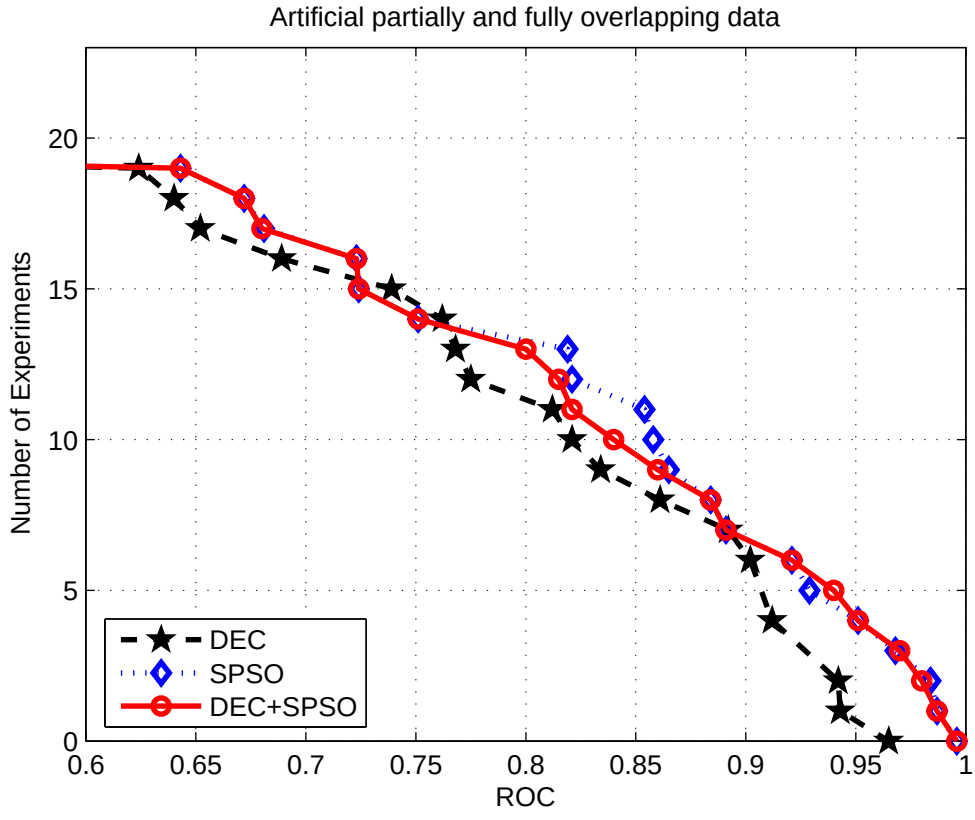
mance of the classifier with the SPSO oversampling technique in fully overlapped datasets is more apparent. When there is more overlapping between the minor and major classes, the problem of imbalanced data is more acute. So the position of the inserted data in the minor class is more important and in our algorithm it has been done more accurately than in SMOTE method. With regard to the time needed for each algorithm, DEC has an advantage compared to our method, because in the oversampling technique the minor class, depending on the number of its instances, is oversampled up to 10 times (in our experiments) which increases the dimension of the of kernel matrix. In contrast, in the DEC method choosing the correct cost of error for minority and majority classes is an important issue. One suggested method is to set the error cost ratio equal to the inverse of the imbalance ratio. But, based on our experiments that value is not always the optimum, and especially in partially and fully overlapped datasets we had instability of performance even with values near the optimal value. Based on our experiments in the well-separated imbalanced data the quality of DEC is very near to the SPSO method and for some experiments, even better, and we could find the optimum value for error cost ratio simply. So perhaps with this kind of datasets one should prefer the DEC method. But with partially and fully overlapping data, we found that our oversampling method in general has better performance, and if it is used along with the DEC method, it not only increases the performance of the classifier but it also makes finding the value for the error cost ratio simpler. We also have more stability with values close to the optimum value of the error cost ratio. The graphs in Fig. 4.7.**a** and Fig. 4.7.**b** show the value of the ROC score of the classifier for partially overlapped artificial sequences from the family of 001-002-024 ($001 - 002 - 024'$) when the DEC method and DEC along with SPSO (400% oversampling) were applied. We see that when SPSO oversampling is used we have stability in ROC score values and after the optimum value, the ROC score does not change. The drawback is, that we again have to find the best value for the error cost ratio and the rate of oversampling through the experiment by checking different values, but in less time compared to only the DEC method, because of the stability that was shown in Fig. 4.7.**b**. We used that method for all partially and fully overlapping artificial data (Table 4.1 and 4.2). For each experiment we oversampled data in different rates and selected different values of error cost ratio until we got the best result. The results in Fig. 4.7.**c** show that for those kind of data the ROC scores of SPSO and DEC + SPSO are nearly the same. But in the second method (DEC + SPSO), we need to oversample data less than in the SPSO only method and we could find the best value of the error cost ratio sooner than in DEC only. With less rate of oversampling in SPSO we get less accurate results but we can compensate that with DEC.

4.5.2 GPCRs Families Classification Results

We used our oversampling technique in classification all GPCRs families at subfamily and sub-subfamily level (mostly we have a low number of sequences). In subfamily classification we randomly partitioned the data in two non-overlapping sets and used a two-fold



a) b)



c)

Figure 4.7: The ROC score at different error cost ratios for artificial sequences of $001 - 002 - 024'$ in (a) classifier with the DEC method and (b) classifier with DEC + SPSO methods (400% oversampled). (c) Comparison of DEC, SPSO and DEC+SPSO methods for imbalanced data. The graph plots the total number of experiments of partially and fully overlapped imbalanced artificial data for which a given method exceeds an ROC score threshold.

Partially overlapping classes- ROC scores

minority class	# of sequences	SMOTE	DEC	SPSO
$001-002-015'$	16	0.863	0.943	0.951
$001-002-016'$	122	0.821	0.912	0.929
$001-002-017'$	68	0.854	0.892	0.884
$001-002-018'$	74	0.912	0.871	0.891
$001-002-020'$	86	0.972	0.975	0.984
$001-002-021'$	40	0.695	0.739	0.723
$001-002-022'$	44	0.725	0.762	0.751
$001-002-023'$	48	0.965	0.982	0.996
$001-002-024'$	8	0.845	0.834	0.865
$001-002-025'$	10	0.945	0.972	0.987
overall ROC-score		**0.859**	**0.882**	**0.896**

Table 4.1: ROC scores obtained on the partially overlapping classes created from peptide families of GPCR dataset, by various methods. DEC = different error cost;

cross validation protocol. The training and testing was carried out twice using one set for training and the other one for testing. To compare with the results of other researchers, the prediction quality was evaluated by Accuracy (Acc), Matthew's correlation coefficient (MCC), and also overall Accuracy ($\overline{Acc}$) and overall MCC ($\overline{MCC}$) as follows:

$$\overline{Acc} = \sum_{i=1}^{N} \frac{Acc(i)}{N} \tag{4.8}$$

$$\overline{MCC} = \sum_{i=1}^{N} \frac{MCC(i)}{N} \tag{4.9}$$

where

$$Acc. = \frac{TP+TN}{(TN+FN+TP+FP)} \tag{4.10}$$

$$MCC. = \frac{TP \times TN - FN \times FP}{\sqrt{(TN+FN)(TP+FN)(TN+FP)(TP+FP)}} \tag{4.11}$$

(TP = number of true positives, TN = number of true negatives, FP = number of false positives , FN = number of false negatives, N=number of subfamilies or sub-subfamily)

Tables 4.5.2, 4.4 and 4.5 show the results of subfamily classification for classes A, B and C of GPCRs. We see that even when the number of sequences is low, the accuracy of our method is high. The overall accuracy for families A, B and C is 98.94%, 99.94% and

Fully overlapping classes- ROC scores

minority class	# of sequences	SMOTE	DEC	SPSO
$001-002-015"$	32	0.673	0.680	0.724
$001-002-016"$	244	0.753	0.775	0.821
$001-002-017"$	136	0.672	0.652	0.643
$001-002-018"$	148	0.591	0.624	0.672
$001-002-020"$	172	0.763	0.821	0.858
$001-002-021"$	80	0.632	0.689	0.681
$001-002-022"$	88	0.615	0.812	0.854
$001-002-023"$	96	0.912	0.942	0.968
$001-002-024"$	16	0.716	0.768	0.819
$001-002-025"$	20	0.908	0.902	0.921
overall ROC-score		**0.723**	**0.766**	**0.796**

Table 4.2: ROC scores obtained on the Fully overlapping classes created from peptide families of GPCR dataset by various methods.

96.95%, respectively, and overall MCC for families A, B and C is 0.98, 0.99 and 0.91, respectively. The results show that almost all of the subfamilies are accurately predicted with our method. At the subfamily level we compared our method with that of Bhasin et al. [22]. They used an SVM-based method with dipeptide composition of protein sequences as input. The accuracy and MCC values of our method outperform theirs. For example in classification of subfamily A, the overall accuracy and MCC of their method were 97.3% and 0.97 but ours are 98.4% and 0.98, respectively. They did a comparison with other previously published methods like that of Karchin et al. [74] and showed that their method outperformed the others.

For sub-subfamily classification we used 5-fold cross validation. Table 4.6 shows the results for the sub-subfamily level. We see that in this level also the accuracy is high and we could classify most of GPCRs sub-subfamilies. We could obtain an overall accuracy of 97.93% and a MCC of 0.95 for all sub-subfamilies. At this level we could increase the accuracy, especially when the number of sequences in the positive training data was less than 10, and there was no example in which with our oversampling method the accuracy decreases.

To the best of our knowledge there is only one study which has been done for sub-subfamily classification [63] in GPCRs families. Their approach is based on bagging a classification tree and they achieved 82.4% accuracy for sub-subfamily classification, which is less accurate than ours (97.93% with MCC of 0.95) despite the fact that they had excluded families with less than 10 sequences (we only excluded families with less than 4 sequences). We think our oversampling technique can be widely used for other applications of protein classification with the problem of imbalanced data and it can be used along with the different error cost (DEC) method to overcome the problem of imbalanced data for protein data.

Class A subfamilies	Accuracy (%)	MCC
Amine	99.9	0.99
Peptide	97.8	0.97
Hormone protein	100.0	1.00
(Rhod)opsin	99.6	0.99
Olfactory	99.9	0.99
Prostanoid	99.9	0.98
Nucleotide-like	100.0	1.00
Cannabinoid	100.0	1.00
Platelet activating factor	100.0	1.00
Gonadotropin-releasing hormone	100.0	1.00
Thyrotropin-releasing hormone	100.0	1.00
Melatonin	100.0	1.00
Viral	87.0	0.80
Lysosphingolipid	100.0	1.00
Leukotriene	100.0	1.00
Overall	98.4	0.98

Table 4.3: The performance of our method in GPCRs subfamily classification (Class A).

4.6 Conclusion

GPCRs family classification enables us to find the specificity for ligands that bind to the receptor and also to predict the function of GPCRs. Our aim in this study was to develop an accurate method for classification of GPCRs at the sub-subfamily level, at which we have the problem of imbalanced data. We chose the local alignment kernel (LA kernel) as a suitable kernel for our classification task. Compared with HMMs, the LA kernel takes more time during the training phase, but according to results of other researchers, the accuracy of discriminative methods with that kernel is higher than with a generative method like HMMs. To solve the problem of imbalance, we suggested a new approach of oversampling for the imbalanced protein data in which the minority class in the data space is oversampled by creating synthetic protein sequences, considering the distribution of the minor and major classes. This method can be used for protein classification problems and remote homology detection, where classifiers must detect a remote relation between unknown sequences and training data with an imbalance problem. We think that this kind of oversampling in kernel-based classifiers not only pushes the class separating hyperplane away from the positive data to negative data but also changes the orientation of the hyperplane in a way that increases the accuracy of classifier. We developed a systematic study using GPCRs as a set of real and artificially generated datasets to show the efficiency of our method and how the degree of class overlapping can affect class imbalance. The results show that our SPSO algorithm outperforms other oversampling techniques. In this chapter, we also presented evidence suggesting that our oversampling technique can be used along with DEC to increase its sensitivity and stability.

Class B subfamilies	Accuracy (%)	MCC
Calcitonin	100.0	1.00
Corticotropin releasing factor	100.0	1.00
Glucagon	100.0	1.00
Growth hormone-releasing hormone	100.0	1.00
Parathyroid hormone	100.0	1.00
PACAP	100.0	1.00
Secretin	100.0	1.00
Vasoactive intestinal polypeptide	100.0	1.00
Diuretic hormone	99.1	0.91
EMR1	100.0	1.00
Latrophilin	100.0	1.00
Brain-specific angiogenesis inhibitor	100.0	1.00
Methuselah-like proteins (MTH)	100.0	1.00
Cadherin EGF LAG (CELSR)	100.0	1.00
Overall	≈ 100	0.99

Table 4.4: The performance of our method in GPCRs subfamily classification (Class B).

Class C subfamilies	Accuracy (%)	MCC
Metabotropic glutamate	92.1	0.84
Calcium-sensing like	94.2	0.82
Putative pheromone receptors	98.7	0.93
GABA-B	100.0	1.00
Orphan GPRC5	97.1	0.96
Orphan GPRC6	100.0	1.00
Taste receptors (T1R)	97.2	0.81
Overall	96.95	0.91

Table 4.5: The performance of our method in GPCRs subfamily classification (Class C).

Class A	subfamilies	Overall Accuracy (%)	Overall MCC
	Amine	97.1	0.91
	Peptide	99.9	0.93
	Hormone protein	100.1	1.00
	(Rhod)opsin	96.6	0.95
	Olfactory	98.9	0.92
	Prostanoid	98.0	0.94
	Gonadotropin-releasing hormone	96.1	0.93
	Thyrotropin-releasing hormone	91.2	0.94
	Lysosphingolipid	98.4	1.00
Class B	Latrophilin	100.0	1.00
Class C	Metabotropic glutamate	98.1	0.96
	Calcium-sensing like	97.2	0.93
	GABA-B	100.0	1.00
Overall		97.93	0.95

Table 4.6: The performance of our method in GPCRs sub-subfamily classification for Class A, B and C.

Chapter 5

Time series Kernels for Biosonar Data Classification

5.1 Introduction

Time series are an important type of data occurring in many scientific disciplines. A common task with time series is to compare one sequence with another. In some domains, a very simple distance method measure, such as Euclidian distance, will suffice. In the case that two time series have similar parts but not at similar positions, we use a more efficient method for similarity extraction known as Dynamic Time Warping (DTW), in which the time of one (or both) sequences is "warped" before an alignment. Dynamic programming is used to measure the similarity score in DTW.

Kernel methods have also been used as a popular method to extract the similarity. Different kernels correspond to different notions of similarity. As we explained in chapter 2, a kernel function implicitly defines a feature space which in many cases we do not need to construct explicitly. The structure of the data and our knowledge of the particular time series suggest a way of comparison that we can consider in our kernel function. Then, the kernel function can be used directly in Support Vector Machines (SVMs) based classifiers.

There have been methods proposed to embed the time alignment operation and DTW into a kernel function [116, 39]. These methods especially are useful in speech recognition, in which the information lies in the whole time series.

However, in some time series the information lies in a fixed (or not very varied) size window of time events (subsequence), independent of the actual time. So we have subsequences at random positions whose similarities should be measured. Those repetitive parts may occur in speech, musical pieces and sonar signals. Therefore, the algorithms for finding similar time series should not consider the whole time series but look for informative subsequences. Then, in kernel based methods for similarity extraction, we need kernels which can extract similarities between all subsequences. Hence, the main task is to find a map that reflects the suitable and common features of those time series and gives a good indiction of the sub similarity we would like to capture. On the other hand, we

should be able to calculate those inner products efficiently.

In this chapter, we study the classification of biosonar signals as an example of the random process signals which contain those local similarities.

Bats can distinguish objects by emitting a series of ultrasound signals (chirps) that generally sweep covering frequencies from 22 to 100 kHz [94]. Despite the similarity between the process of analysis of reflected echoes in bats and that of the hearing system in human beings, it is difficult for us to understand the process, because we have never explored the environment with echolocation.

To unravel the mechanism of echolocation, inspired by the bat biosonar system, researchers have utilized biosonar heads and ultrasonic sensing techniques similar to that of bats for mobile robots (biomimetic robots) and tried to classify different textures and landmarks through their received echo signals [91, 82].

McKerrow used a CTFM (Continuous Transmission Frequency Modulated) system and modelled the echoes with the acoustic density profiles, used the frequency components and energy spectra and found features, which characterize the acoustic density profiles of plants in a classification task [91]. Kuc [82] suggested a transformation of echo to pseudo-action potential as a temporal point process to understand how bats recognize landmarks in the field. Müller [97] presented a neuro-spike representation of echoes in which each echo is transcribed into a spike code using a parsimonious model, and classified four foliages using three features derived from interspike intervals. Gao et. al [48] presented a template matching algorithm for classification of several types of brick walls, picket fences and hedges using sonar echoes. M. Wang et al. [134, 135] used different structural features in the frequency domain and also template matching for the classification task.

We used a Biosonar–based robot and ultrasound signals (chirps) and simulated echoes of the bat while using different trees as landmarks (Fig. 5.4). By comparing the returning echoes (which are individually the superposition results of the reflected echoes) and pattern recognition methods, we aim at recognizing the objects with emphasis on understanding and exploiting the characteristics of bat sonar system. From the study of works of the researchers noted above [91, 82, 97, 48] and also our experiments, we concluded that finding robust feature for classification is not trivial. For example, the orientation of objects can result in large changes in the reflected echoes. Hence, in this case features which are only temporally based can be inefficient. But on the other hand, the local temporal similarities between different echoes of one object as an indication of its texture is a significant issue that should be considered.

Our approach is a combination of system neurobiology with sonar signal processing and pattern recognition to learn how bats process echoes and perceive objects.

We develop an efficient method for our classification task and consider both local temporal similarity and the power spectrum of echoes and propose a kernel based classification method considering those parameters.

We suggest a kernel called *time-resolved spectrum kernel* to measure the similarity of echoes as time series. The p-length subsequence of that kernel simply measures the

occurrences of fixed p-length subsequences for each of the time series in consideration. The more time series share similar p-length subsequences, the more similar they are.

We also implement a more general kernel called *warped time-resolved spectrum kernel*, which considers warping in the subsequences. The warped time-resolved spectrum kernel measures the whole similarities of all warped non contiguous subsequences of the two time series, independent of their positions. We then use those kernels directly in a SVM-based classifier. The results show that those kernels allow for a very reliable discrimination of reflected sonar echoes from different objects.

This chapter is organized as follows: In the next section we begin by illustrating the echolocation and biosonar systems. In section 5.3 we explain the main parts of a biosonar based robot. In sections 5.4 and 5.5, we discuss the time-resolved spectrum kernel, and the results are given in section 5.6. At last, we conclude our work in section 5.7.

5.2 Echolocation and Biosonar

Vision and audition are close phenomena in that both can process reflected waves of energy. Vision processes photons (waves of light) as they travel from their source, bounce off surfaces throughout the environment, and enter the eyes. In fact, the visual system is able to perceive its surrounds by its ability to process the complex patterns of photons of visible light as they reflect into the eye from surfaces in those surroundings. If all one could see were sources of light and not reflected light, our eyes would give us very little awareness of the nature of our surroundings. By perceiving and interpreting patterns of reflected light, extremely rich and detailed information can be gathered about the layout and characteristics of surrounding space and objects therein.

Similarly, the auditory system can process phonons (waves of sound), reflected from their source, bounce off surfaces, and enter the ears. The auditory system then can extract a great deal of information about the environment by interpreting the complex patterns of reflected energy that they receive. Echo information can be perceived and processed by the auditory system to enable many determinations about surrounding space and one's physical relationship to it.

Echolocation, the sonar 'sight' of bats, is similar to the "SOund NAvigating and Ranging" or sonar used by the military. Because it is produced by living organisms rather than by machines, it is often called 'biosonar'. The term echolocation was first coined by Donald Griffin in 1938 [52], who discovered that bats navigate with the aid of high frequency sounds bouncing off obstacles in their environment. It is an aspect of auditory perception which may be broadly defined as the ability to perceive echoes. Echolocation makes it possible for species to decrease their dependence on the visual system; such independence confers advantages to the echolocator for navigation and hunting under poor lighting conditions.

Numerous investigations such as those concerning by bats, nocturnal birds, and marine animals [7, 53] clearly demonstrate that echoes can provide detailed and consistent

information about the surrounding environment. With this information, sightless animals perform all essential functions of productive living similar to those with sight.

Such studies of echolocation may be of great value to blind people by making available the knowledge needed to improve nonvisual competence in spatial awareness and travel. A thorough understanding of the nature of this skill could have valuable implications for training and rehabilitation. In [120] possible uses of echolocation by humans are discussed and it is argued that echolocation may be a basic perception–action ability of humans.

Some simple and accurate examples conducted by [53, 110, 111] lead us to a comprehensive and practical understanding of the processes behind echolocation and its utility in human while suggesting that both blind and sighted humans are capable of substantial precision in the perception of properties of distal objects, such as distance, size, shape, substance, and relative motion by echolocation.

Inspired by bat echolocation research, the human echolocation study was undertaken in the hopes of acquiring a more intimate knowledge about human echolocation ability. Findings demonstrate the ability of blind humans to use echolocation to actively seek out objects in their vicinity and thus to exert more control over perceiving the qualities of objects in their environment. Echolocation may in fact be a tool for the blind to perceive, not just the presence of objects, but such dimensions of the objects as size and distance.

Bats can distinguish objects and their prey by emitting a series of ultrasound signals (chirps) that generally sweep covering frequencies from 22 to 100 kHz. Lie et al. [83] point out that certain species of bats can use echoes elicited by their own ultrasonic chirps can perceive obstacles as thin as 0.65 mm. These authors further indicate that some echolocating bats can develop a precise spatial memory of previously explored environments to an accuracy within 2 centimeters and resolve reflecting points as close together as 0.3 mm in range. The acoustic image of a sonar target is apparently derived from time-domain or periodicity information processing by the nervous system.

The bat has a sonar transmitter (mouth) and two sonar receivers (part of its auditory system) which it uses to receive and analyze echoes reflected from targets in the environment. It emits short high-bandwidth clicks in a forward focused beam, and listens for reflected echoes. During this process, the animal gathers useful information about the insonified targets. These pulses are usually frequency modulated (FM), constant frequency (CF) or combinations of both [83].

5.3 Biosonar based robot

5.3.1 Hardware

The implementation of the whole system consists of a mobile robot (Robin) with two PCs, a digital signal processing package, and a biosonar system (Fig. 5.1). The biosonar system includes a National Instruments NI6110 analog I/O card, a mini servo controller

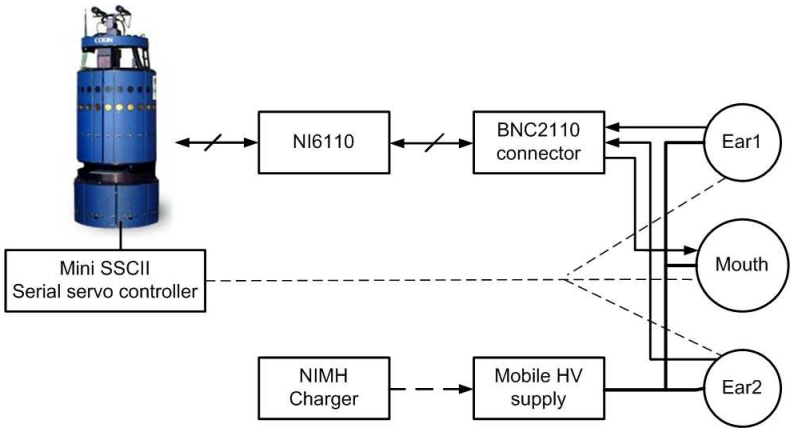


Figure 5.1: Biosonar system configuration

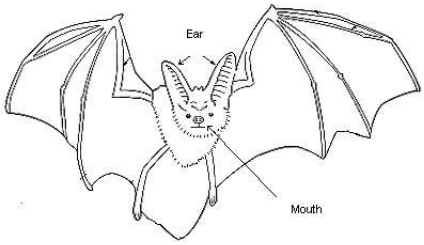
(module SSCII), a BNC2110 connector, and the biosonar head. The NI6110 card and the BNC2110 connector transfer chirp signals and receive the reflected echoes. The biosonar head (Fig. 5.3) consists of 3 Polaroid sensors in a triangular layout, similar to the layout of a bat's mouth and ears: two Polaroid 600 sensors spaced 12.5 cm apart as *ears*, a Polaroid 7000 sensor as *mouth* in the middle between two ears. Each of the two ears has two degrees of angular freedom provided by two servo motors. These can be finely rotated to acquire local support. The Polaroid ultrasonic ranging system is most commonly used by the robotics research community.

The maximum sampling speed of the NI6110 card is 5 MHz. We utilized 1 MHz in our research. The NiMH charger box provides the sensors with a 150V power supply. The mobile robot Robin is an autonomous mobile service robot that has two PCs inside, one is in charge of navigation control, the other one is responsible for signal data processing, feature extraction and decision making.

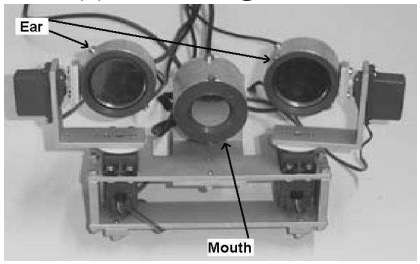
5.3.2 Chirp Design

Bats utilize many different types of echoes depending upon whether they are hunting, flying in a densely forested area, or flying in an open space [122]. Some bats use brief, broadband, frequency modulated (FM) calls, while others emit more prolonged, constant frequency (CF) calls. CF chirps are more suitable for detection than for tracking. But in FM chirps, the frequency changes with its duration and it consists of several modulated FM components. It closely resembles a radar's chirp signal and lends itself well to range finding. Furthermore, it yields a spectral signature that is useful for determination of an object's size, shape and surface detail and discrimination between object types. Some bats have developed nonlinear frequency modulation too.

CF–FM bats can switch between waveforms. They employ CF and FM type waveforms during a single engagement and for example use CF pulses when looking for prey



(a) The biological bat



(b) Biosonar head

Figure 5.2: The biosonar head consists of one emitter (mouth) and two receivers (ears).

in a stationary position but switch to a CF–FM signal once they are tracking a target.

Considering the task of our research – natural landmark classification – we used the FM chirp with amplitude adaptation, which means the frequency sweeps linearly in a range and the amplitude varies in an oval form. It resembles the chirp form of most bats in nature.

In our experiments, the emitted pulse was a linearly frequency modulated chirp sweeping from 20kHz to 120kHz in 1 ms (Fig.5.3).

5.3.3 Landmarks and Sensing Strategy

Through echolocation in darkness, a bat can perceive not only the position of an object, but also its 3D structure [54]. The recognizable target in nature works as a landmark for its navigation. For our sensory task, these landmarks should be rich and easy to be found there. The criteria for selecting natural landmarks include observability, frequent occurrence, uniqueness, temporal stability, easy classification, and lateral compactness [107]. Considering those aspects, we selected three artificial trees with similar height of 1.7 m as shown in Fig. 5.4.

Compared with other researchers [97, 82], we used a different method for sensing the objects. We used a 0.5 degree angular stepsize for our scans, each tree was scanned 360 degrees in a circular movement of the robot and we collected echoes from all orientations of leaves and tree. The reflected echo contains the information about the geometry of the tree and is the superposition of all reflections [134, 135].

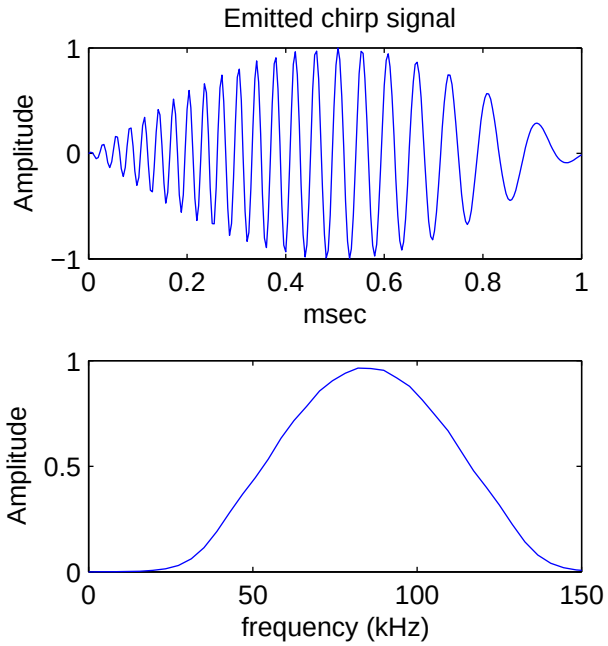


Figure 5.3: Emitted chirp signal and its frequency content.

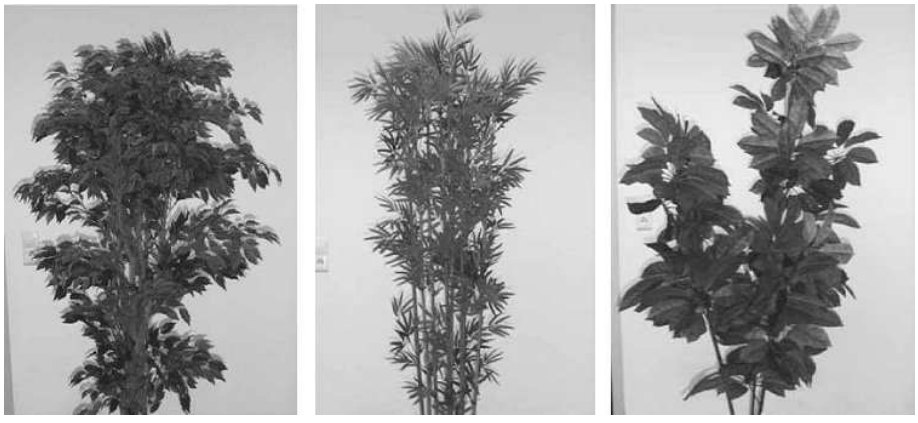


Figure 5.4: Three different trees as biosonar landmarks. From left to right: Ficus, Bamboo, Schefflera.

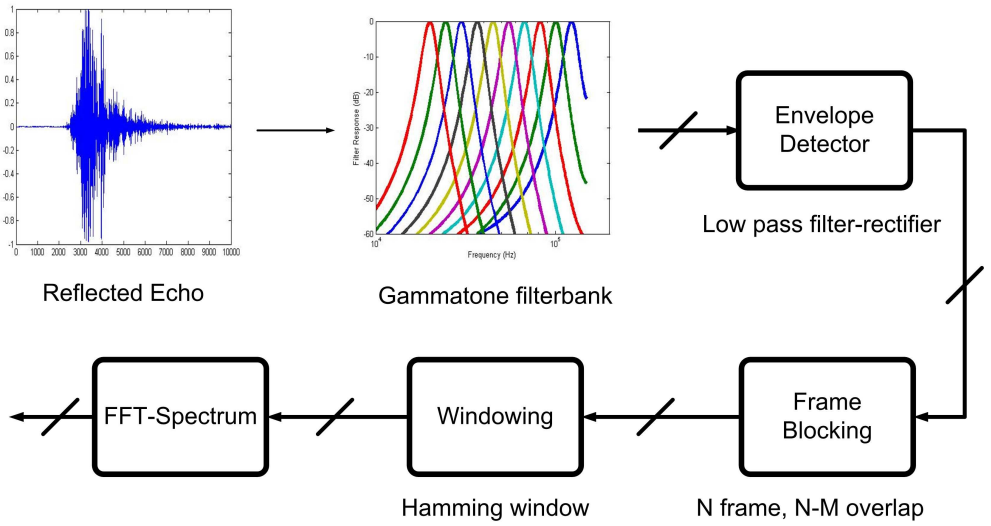


Figure 5.5: Block diagram of the preprocessing steps for reflected echoes.

5.3.4 Data Processing

Fig. 5.5 shows the block diagram of the data acquisition and preprocessing procedure of reflected echoes. We passed the reflected echoes through a bank of 10 gammatone filters between 20 kHz and 120 kHz. In order to extract the envelope of the filtered signals, they were delivered to half-wave rectifiers.

The next step is *frame blocking*. In this step the signal blocked to frames of N samples, is separated from adjacent frames by M $(M < N)$ samples and has $N-M$ overlaps. Considering the sampling frequency of the data acquisition part (1 MHz) and the minimum width of leaves of trees and axial resolution of transducers, we selected $N = 32$ and 50% overlap for frames. The next step in the data preprocessing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. We used a *Hamming window* for this purpose. The last step is to calculate the average energy of each band of gammatone filter bank in each frame. The result is a feature matrix, where each column is a vector showing the average energy of each channel in one time frame. Fig. 5.7 shows the examples of the preprocessed reflected echoes from Ficus, Bamboo and Schefflera trees. We use this feature matrix for our classification task.

After the preprocessing steps for each echo (Fig. 5.5), we have a matrix of time series in which each cell is a time frame and its value is the average energy of each channel of gammatone filter. Furthermore we have

$$A = C \times S \tag{5.1}$$

where A is the number of features, C is the number of channels and S the number of samples in each channel.

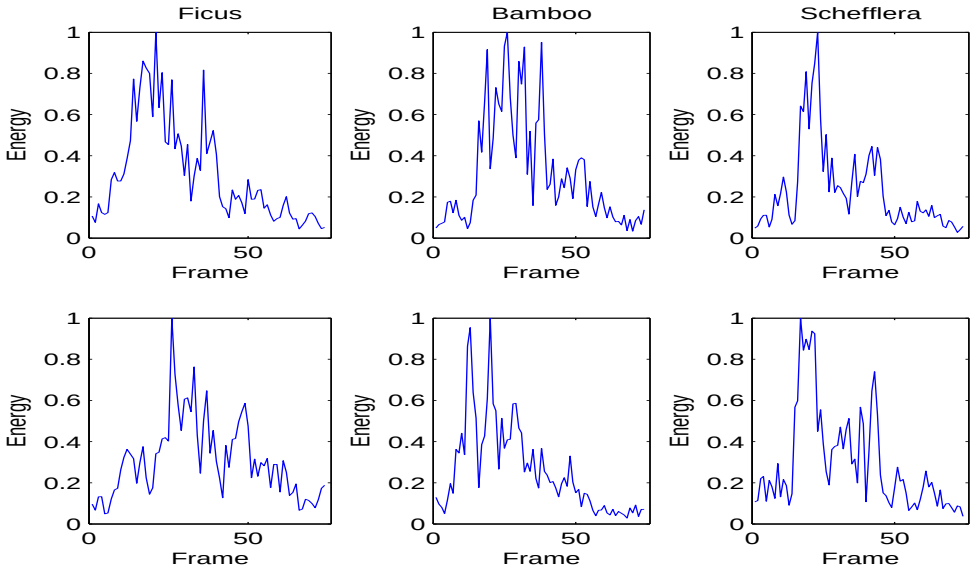


Figure 5.6: Features: examples of the energy spectrum (output of gammatone filter centered around 50 kHz) for Ficus, Bamboo and Schefflera trees.

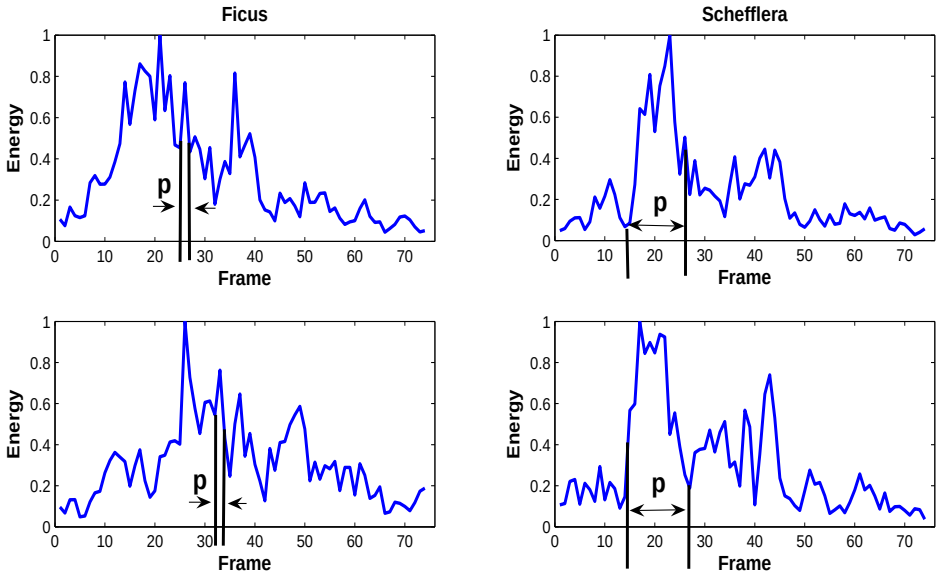


Figure 5.7: The energy spectrum in each time frame for Ficus and Schefflera trees (output of gammatone filter centered around 50 kHz). The time-resolved spectrum kernel tries to find the local similarities in window of size p in echoes of one object.

Fig. 5.7 shows the examples of the preprocessed echoes of Ficus and Schefflera trees. We use those feature matrices for our classification task.

As noted before, the biosonar signals are random and nonstationary in the temporal dimension and small changes in the orientation of the plant result in changes in the position of energy features along the time series. For example, the location of leaves in the plant determines the acoustic energy throughout the frames, and small changes in the orientation of the plant result in changes in those features along the frames of time. But, as we see in Fig. 5.7, despite the seemingly randomness of those signals, there are some local similarities (shown by p) in echoes from one tree. Then, if we can find the sizes of windows in which we have maximum similarity between data of one object, it can help us to classify that object from others. We consider the output of the block diagram shown in Fig. 5.5, a time series, in which each point is a time frame and its value is a vector of features (the average energy of each channel of gammatone filter bank). We should find the subsequences of the time series *independent of the positions of occurrences* that have maximum similarities in echoes of each object. The intuition behind our idea is that the structure of objects and, as an example, the size of leaves or branches, should be considered in the classification task. The size of the subsequence that we are looking for, can be related to the size of the leaves or branches of the tree. In another way, the energy reflected by the leaves or branches of the tree can be related to the size of those similar subsequences of the time series.

A similar situation happens in text classification and also remote homology detection in protein families, where we must detect a remote relation between an unknown sequence and a family of proteins. Those proteins contain domains whose positions are not similar in proteins of a family. There again we should measure the local similarities between all subsequences as an indication of similarity between two sequences.

Similar to the remote homology detection in proteins, where a classifier must detect a remote relation between an unknown sequence and a family of proteins, in our classification task, the algorithms for finding similar time series should not consider the whole time series but look for informative subsequences, and we need kernels, which can extract similarities between subsequences.

Inspired by the solutions for remote homology detection in protein families and the string kernel proposed by Lodhi et al. [90], we re–implement the spectrum kernel algorithm for time series and suggest a kernel called *time-resolved spectrum kernel* to measure the similarity of two time series [10, 14]. The p-length subsequence of that kernel simply measures the occurrences of fixed p-length subsequences for each of the time series in consideration, independent of their positions. Then, we implement a more general kernel called *warped time-resolved spectrum kernel*[14], which considers warping in the subsequences. The warped time-resolved spectrum kernel measures the whole similarities of all warped non contiguous subsequences of the two time series, independent of their positions.

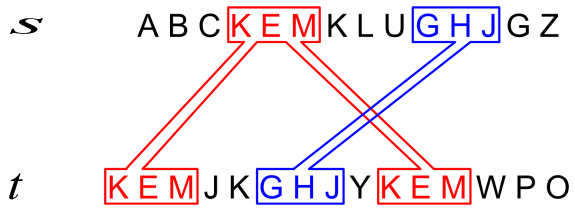


Figure 5.8: The p–spectrum kernel counts the common p–length subsequences between two strings (here p=3).

5.4 Time-resolved Spectrum Kernel

A kernel function can often be considered as a measure of similarity. Different kernels correspond to different notions of similarity. The structure of the data and our knowledge of the particular time series suggest a way of comparison that we can consider in our kernel function. The use of a kernel makes it possible to perform the mapping into that feature space and to calculate the inner product between those maps. But the main task here is to find a map ϕ that reflects the suitable and common features of those time series and gives a good indication of the similarity we would like to capture.

p–spectrum is an efficient sequence–similarity kernel, proposed by Lesli et al. [85], which counts the common fixed length subsequences between two strings.

Definition 5.1 (p–spectrum) *Given a number $p \geq 1$, the p–spectrum of a string is the set of all the p–length contiguous subsequences that it contains.*

In a spectrum kernel, the feature space F is indexed by all the subsequences u from the alphabet Σ and its elements count the number of times a p–*mer* occurs in the sequence. For $u \in \Sigma^p$, the implicit embedding map ϕ brings s to F:

$$\phi : s \rightarrow (\phi_u(s)) \in F$$

where

$$\phi_u(s) = \text{number of times } u \in \Sigma^p \text{ occurs in } s$$

then the p–spectrum kernel between strings s and t is the inner product in the feature space:

$$\mathcal{K}_p(s,t) = \langle \phi_p(s), \phi_p(t) \rangle$$

Figure 5.8 and table 5.1 show an example of a 3-spectrum kernel of s and t. We see that the more common substrings two strings have, the larger is the kernel value and so the more the two strings are similar.

Table 5.1: 3-spectrum kernel between s and t in Fig. 5.8, $\mathcal{K}_3(s,t) = \langle \phi_3(s), \phi_3(t) \rangle = 2 \times 1 + 1 \times 1 = 3$.

ϕ	KEM	GHJ
s	1	1
t	2	1

Similar to the above p–spectrum kernel for strings, the time-resolved spectrum kernel simply measures the whole similarities of all subsequences of the time series in consideration, independent of their positions. The more two time series share similar subsequences, the more similar they are.

A time sequence $s = s_1 ... s_n$ is a sequence of data points at successive times with $s_i \in \mathbb{R}^d$, where $1 \leq i \leq n$ and d is the dimension of data points. we denote $|s|$ the length of s and $s(i-p+1:i)$ the p-length subsequence of s from position $i-p+1$ to position i.

Definition 5.2. *We denote $\mathbf{I}_p^{|s|}$ the set of indices, defining all the p-long contiguous subsequences of s*:

$$\mathbf{I}_p^s = \{\mathbf{i} : \mathbf{i} \in \mathrm{N}^p, 1 \leq i_1 < ... < i_p \leq |s|\}$$

and $s_\mathbf{i}$ is a subsequence of s in positions given by $\mathbf{i} = (i_1, i_2, ..., i_p)$.

For $u \in \Sigma^{p \times d}$, the infinite set of all subsequences with size p and dimension d, the implicit embedding map ϕ brings s to vector space F, $\phi : s \rightarrow (\phi_u(s)) \in F$. The u component of our feature vector is defined as:

$$\phi_u^p(s) = \sum_{\mathbf{i} \in \mathbf{I}_p^s,\ u \in \Sigma^{p \times d}} \varphi_u(s_\mathbf{i})$$

where φ is an implicit map that satisfies:

$$\kappa_p(s_\mathbf{i}, t_\mathbf{j}) = < \varphi_u(s_\mathbf{i}),\ \varphi_u(t_\mathbf{j}) > ,\ \text{for} \quad \mathbf{i} \in \mathbf{I}_p^s,\ \mathbf{j} \in \mathbf{I}_p^t \tag{5.2}$$

in which κ_p is a valid kernel function that measures the similarity between two p-length contiguous subsequences $s_\mathbf{i}$ and $t_\mathbf{j}$ of the time series in consideration. In words, $\phi_u^p(s)$ is a sum over all similarities between p-long subsequences of s and u. The dot product of those feature vectors represents the time resolved p-spectrum kernel (spectrum kernel with subsequence size of p):

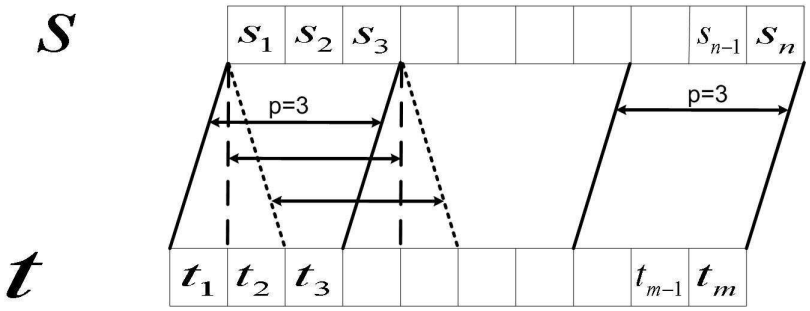


Figure 5.9: The time-resolved p–spectrum kernel adds the similarities of all p–length contiguous subsequences between two time series (here p=3).

$$\begin{aligned}\mathcal{K}_p(s,t) &= \langle \phi_u^p(s), \phi_u^p(t) \rangle = \int_{\mathbf{R}^{d\times p}} \phi_u^p(s)\phi_u^p(t)du \\ &= \sum_{\mathbf{i}\in\mathbf{I}_p^s}\sum_{\mathbf{j}\in\mathbf{I}_p^t} \int_{R^{d\times p}} \varphi_u(s_\mathbf{i})\varphi_u(t_\mathbf{j})du \\ &= \sum_{\mathbf{i}\in\mathbf{I}_p^s}\sum_{\mathbf{j}\in\mathbf{I}_p^t} \kappa_p(s_\mathbf{i}, t_\mathbf{j})\end{aligned}$$

The above equation says that the spectrum kernel is a summation of all subsequences similarities. Considering the definitions of $\mathbf{I}_p^s$ and $\mathbf{I}_p^t$, we can say:

$$\mathcal{K}_p(s,t) = \sum_{i=p}^{|s|}\sum_{j=p}^{|t|} \kappa_p(s(i-p+1:i), t(j-p+1:j)) \tag{5.3}$$

Needless to say, the computation cost of that kernel is high. The evaluation of κ_p requires $O(p)$ computations, and the cost for computation of $\mathcal{K}_p(s,t)$ is of order $O(p|s||t|)$. In string p-spectrum kernels, a very fast method for computation of $\mathcal{K}_p(s,t)$ is to use an efficient data structure known as 'trie' (retrieval tree) in which we build a suffix tree for the collection of p-length subsequences of s and t, obtained by moving a p-length sliding window across each of s and t, and then calculate the kernel by traversing the tree. But because of an infinite subsequence set, that method is not applicable for the time series spectrum kernel unless the time series is quantized, symbolized and converted to a string. In this case we are faced with the quantization errors and the method for quantization and symbolization can affect the efficiency of the kernel method. Instead of that we use dynamic programming to calculate the time-resolved spectrum kernel while accepting some

constraints. We accept a constraint on choosing the kernel function $\kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}})$, we suppose:

$$\kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}}) = \prod_{i=1}^{p} \kappa^*(s_{\mathbf{i}i}, t_{\mathbf{j}_i}) \tag{5.4}$$

in which κ^* is an arbitrary function (discussed later) that measures the similarity between two data points. Considering Equations 5.4 and 5.3, we define an auxiliary kernel, p-suffix kernel $\mathcal{K}_p^S(s', t')$ as:

$$\begin{aligned} \mathcal{K}_p^S(s', t') &= \begin{cases} \kappa_p(s'(|s'|-p+1{:}|s'|), t'(|t'|-p+1{:}|t'|)) & \text{if } \min(|s'|,|t'|)\geq p \\ 0 \quad \text{otherwise.} \end{cases} \\ &= \begin{cases} \prod_{i=0}^{p-1} \kappa^*(s'_{|s'|-i}, t'_{|t'|-i}) & \text{if } \min(|s'|,|t'|)\geq p \\ 0 \quad \text{otherwise.} \end{cases} \end{aligned} \tag{5.5}$$

$$\tag{5.6}$$

where $s' = s(1 : |s'|)$, $t' = t(1 : |t'|)$, $1 \leq |s'| \leq |s|$ and $1 \leq |t'| \leq |t|$. Then we express the p-spectrum kernel in terms of its suffix version as:

$$\mathcal{K}_p(s', t') = \sum_{i=1}^{|s'|} \sum_{j=1}^{|t'|} \mathcal{K}_p^S(s'(1 : i), t'(1 : j)) \tag{5.7}$$

If we add a new data point x to the time series s', using the above equation we can calculate $\mathcal{K}_p(s'x, t')$:

$$\begin{aligned} \mathcal{K}_p(s'x, t') &= \sum_{i=1}^{|s'x|} \sum_{j=1}^{|t'|} \mathcal{K}_p^S(s'x(1 : i), t'(1 : j)) \\ &= \sum_{i=1}^{|s'|} \sum_{j=1}^{|t'|} \mathcal{K}_p^S(s'(1 : i), t'(1 : j)) + \sum_{j=1}^{|t'|} \mathcal{K}_p^S(s'x, t'(1 : j)) \\ &= \mathcal{K}_p(s', t') + \sum_{j=1}^{|t'|} \mathcal{K}_p^S(s'x, t'(1 : j)) \end{aligned} \tag{5.8}$$

On the other hand, if we add another new data point y to the time series t', considering equation 5.4 and the above definition of $\mathcal{K}_p^S$, we can say:

$$\mathcal{K}_p^S(s'x, t'y) = \kappa^*(x, y)\mathcal{K}_{p-1}^S(s', t') \tag{5.9}$$

It is clear that: $\mathcal{K}_p(s, t) = \mathcal{K}_p(s', t')$ if $s = s', t = t'$. Now, we define a recursive computation for $\mathcal{K}_p$:

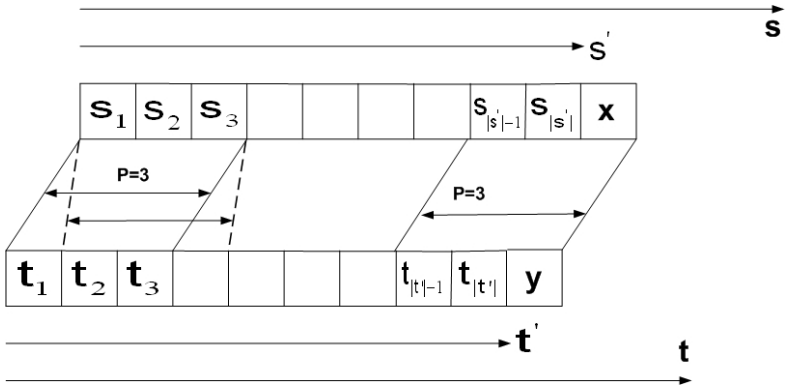


Figure 5.10: The time-resolved kernel is calculated using the suffix kernels and the recursion over prefixes continues until $x = s_{|s|}$ and $|t'| = |t|$

Definition 5.3: *Recursive computation of the time resolved spectrum kernel.*

$$\mathcal{K}_p(s'x, t') = \mathcal{K}_p(s', t') + \sum_{k=1}^{|t'|} \mathcal{K}_p^S(s'x, t'(1:k)) \tag{5.10}$$

$$\mathcal{K}_p^S(s'x, t'(1:k)) = \kappa^*(x, t'_k)\mathcal{K}_{p-1}^S(s', t'(1:k-1)) \tag{5.11}$$

$$\begin{aligned} \mathcal{K}_0^S(s', t') &= 1 \quad for \quad all \quad s', t', \\ \mathcal{K}_i^S(s', t') &= 0, \quad if \ \min(|s'|, |t'|) < i, \\ \mathcal{K}_i(s', t') &= 0, \quad if \ \min(|s'|, |t'|) < i, \end{aligned}$$

The computation of the kernel follows a dynamic programming technique. We have recursions over the prefixes of the time series and the lengths of the subsequences and we do the routine above until $x = s_{|s|}$ and $|t'| = |t|$. Table 5.2 shows the steps for calculation of $\mathcal{K}_p^S$ using $\mathcal{K}_{p-1}^S$ when $p = 3$.

To prevent that with larger sizes of subsequences the kernel achieves a higher similarity score we normalize the kernel:

$$\mathcal{K}_i^{norm}(s, t) = \frac{\mathcal{K}_i(s, t)}{\sqrt{\mathcal{K}_i(s, s)\mathcal{K}_i(t, t)}}$$

This operation scales the similarities in the range [0,1].

As we see from the above pseudo-code, the evaluation of the $\mathcal{K}_i^{norm}$ is of order $O(|s||t|)$ and the overall complexity of our algorithm to calculate a linear combination of all p-spectrum kernels is $O(p|s||t|)$ while if Equation 5.3 is used the complexity is of order $O(p^2|s||t|)$.

Algorithm Time resolved spectrum kernel

Input : Time series s and t of length n and m, max subsequence length l;
Output: Array of spectrum kernel $\mathcal{K}[]$ with different sizes of subsequence-length from 1 to l);

$KPS(0:n, 0:m, 0) = 1$; /* KPS(i, j, p) stores $\mathcal{K}_p^S(s(1:i), t(1:j))$ */
$KP(0:n, 0:m) = 0$; /* KP(i, j) stores $\mathcal{K}_p(s(1:i), t(1:j))$ */
for $p \leftarrow 1$ **to** l **do**
  $KPS(0:n, 0, p) = 0$;
  $KPS(0, 0:m, p) = 0$;
  for $i \leftarrow 1$ **to** n **do**
    $P(0)$=0; /* P(k) stores the second term on the right side in Eq. 5.10 */
    for $k \leftarrow 1$ **to** m **do**
      $KPS(i, k, p) = \kappa^*(s_i, t_k) KPS(i-1, k-1, p-1)$;
      $P(k) = P(k-1) + KPS(i, k, p)$; $KP(i, k) = KP(i-1, k) + P(k)$;
    end
  end
  $\mathcal{K}[p] = KP(n, m)$; /* $\mathcal{K}_p(s, t)$ */
end
return $\mathcal{K}[]$

$KPS(:,:,1) = \mathcal{K}_1^S$	ϵ	$t' = t_1$	$t' = t_1t_2$	$t' = t_1t_2t_3$
ϵ	0	0	0	0
$s' = s_1$	0	$k^*(s_1, t_1)$	$k^*(s_1, t_2)$	$k^*(s_1, t_3)$
$s' = s_1s_2$	0	$k^*(s_2, t_1)$	$k^*(s_2, t_2)$	$k^*(s_2, t_3)$
$s' = s_1s_2s_3$	0	$k^*(s_3, t_1)$	$k^*(s_3, t_2)$	$k^*(s_3, t_3)$

$\mathcal{K}_2^S$	ϵ	$t' = t_1$	$t' = t_1t_2$	$t' = t_1t_2t_3$
ϵ	0	0	0	0
$s' = s_1$	0	0	0	0
$s' = s_1s_2$	0	0	$k^*(s_2, t_2)k^*(s_1, t_1)$	$k^*(s_2, t_3)k^*(s_1, t_2)$
$s' = s_1s_2s_3$	0	0	$k^*(s_3, t_2)k^*(s_2, t_1)$	$k^*(s_3, t_3)k^*(s_2, t_2)$

$\mathcal{K}_3^S$	ϵ	$t' = t_1$	$t' = t_1t_2$	$t' = t_1t_2t_3$
ϵ	0	0	0	0
$s' = s_1$	0	0	0	0
$s' = s_1s_2$	0	0	0	0
$s' = s_1s_2s_3$	0	0	0	$k^*(s_3, t_3)k^*(s_2, t_2)k^*(s_1, t_1)$

Table 5.2: Calculation of $\mathcal{K}_p^S$ using $\mathcal{K}_{p-1}^S$ for $s = s_1s_2s_3$, $t = t_1t_2t_3$ and $\mathcal{K}_0^S = 1$. $\mathcal{K}_3(s,t) = \sum_{i=1}^{3}\sum_{j=1}^{3}\mathcal{K}_p^S(s(1:i)t(1:i))$.

We considered $\kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}})$ (Equation 5.2) as a product of similarities between data points (5.4). Different choices of that function allow different methods of comparing sub similarities. As a suitable selection we consider:

$$\kappa^*(s_{\mathbf{i}i}, t_{\mathbf{j}_i}) = \exp\frac{-(s_{\mathbf{i}i} - t_{\mathbf{j}_i})^2}{2\sigma^2}$$

to measure the similarity between two data points, then:

$$\kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}}) = \prod_{i=1}^{p}\kappa^*(s_{\mathbf{i}i}, t_{\mathbf{j}_i}) = \exp\left(-\frac{||s_{\mathbf{i}} - t_{\mathbf{j}}||^2}{2\sigma^2}\right) \tag{5.12}$$

$\kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}})$ is the gaussian kernel of width σ and suitable for measuring the similarity of subsequences in the time series.

In practice and specially in our classification task, it makes sense to consider the similarity of subsequences having different sizes and calculate a linear combination of differ-

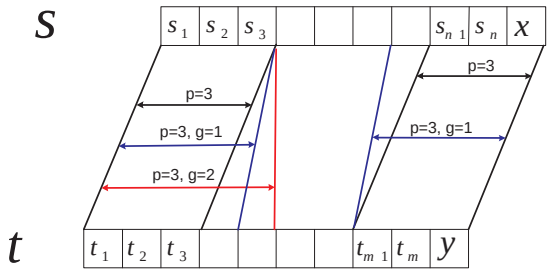


Figure 5.11: The warped time-resolved p–spectrum kernel adds the similarities of all possibly warped p–length subsequences between two time series (here p=3).

ent i-spectrum kernels with different weighting $\theta_i \geq 0$. The weighted kernel is:

$$K(s,t) = \sum_{i=1}^{l} \theta_i \mathcal{K}_i^{norm}(s,t) \tag{5.13}$$

The parameter θ_i shows the weight of each i-length kernel and the optimum selection of those parameters extracts maximum similarities in the signals in consideration.

5.5 Warped Time-resolved Spectrum Kernel

In this section, we implement a more general kernel called *warped time-resolved spectrum kernel*, which considers warping in the subsequences. The warped time-resolved spectrum kernel measures the whole similarities of all warped non contiguous subsequences of the two time series, independent of their positions. Again, the more two time series share similar subsequences, the more similar they are.

In p-length warped time resolved spectrum kernel, we add the similarities of all (possibly warped) p-length subsequences of times series s and t.

Definition 5.4. *We denote* $\mathbf{I}_p^{|s|}$ *the set of indices, defining all the* p*-long* both *contiguous and non–contiguous subsequence of* s:

$$\mathbf{I}_p^s = \{\mathbf{i} : \mathbf{i} \in \mathrm{N}^p, 1 \leq i_1 < ... < i_p \leq |s|\}$$

and $u = s_{\mathbf{i}}$ *is a subsequence of* s *in positions given by* $\mathbf{i} = (i_1, i_2, ..., i_{|u|})$*. The number of gaps in the subsequence is* $g_{\mathbf{i}} = (i_{|u|} - i_1 + 1) - |\mathbf{i}|$.

For example, if we consider $s = s_1s_2s_3s_4s_5$, $u = s_1s_3s_5$ is a subsequence of s in the positions $\mathbf{i} = (1, 3, 5)$ of length $|\mathbf{i}| = 3$ and $g_{\mathbf{i}} = 2$.

For $u \in \Sigma^{p\times d}$, the set of all subsequences with size p and dimension d, the implicit embedding map ϕ brings s to a vector space F ($\phi : s \rightarrow (\phi_u(s)) \in F$) and the u component of our feature vector is:

$$\phi_u^p(s) = \sum_{\mathbf{i}\in\mathbf{I}_p^{|s|},\ u\in\Sigma^{p\times d}} \varphi_u(s_{\mathbf{i}})\gamma^{g_{\mathbf{i}}}$$

where $\gamma \in (0, 1)$ is a decay factor as a cost for warping (non-contiguousity) in the time series and φ is an implicit map that satisfies:

$$\kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}}) =< \varphi_u(s_{\mathbf{i}}),\ \varphi_u(t_{\mathbf{j}}) > \qquad \mathbf{i} \in \mathbf{I}_p^s,\ \mathbf{j} \in \mathbf{I}_p^t,\ u \in \Sigma^{p\times d} \tag{5.14}$$

in which κ_p is a kernel function that measures the local similarity between two p-length subsequences $s_{\mathbf{i}}$ and $t_{\mathbf{j}}$ of the time series in consideration. In words, $\phi_u^p(s)$ is a sum over all similarities between p-long subsequences of s and u. The dot product of those feature vectors of s and t represents the *warped time resolved p-spectrum kernel*:

$$\begin{aligned}\mathcal{K}_p(s,t) &= \langle\phi_u^p(s), \phi_u^p(t)\rangle = \int_{\mathbf{R}^{d\times p}} \phi_u^p(s)\phi_u^p(t)du \\ &= \sum_{\mathbf{i}\in\mathbf{I}_p^s}\sum_{\mathbf{j}\in\mathbf{I}_p^t} \gamma^{g_{\mathbf{i}}}\gamma^{g_{\mathbf{j}}} \int_{R^{d\times p}} \varphi_u(s_{\mathbf{i}})\varphi_u(t_{\mathbf{j}})du\end{aligned}$$

Regarding the above kernel definition for local similarity (Eq. 5.14), we conclude:

$$\mathcal{K}_p(s,t) = \sum_{\mathbf{i}\in\mathbf{I}_p^s}\sum_{\mathbf{j}\in\mathbf{I}_p^t} \kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}})\gamma^{g_{\mathbf{i}}+g_{\mathbf{j}}} \tag{5.15}$$

As we see from the above equation, the kernel adds all similarity scores between subsequences, considering all possible degrees of warping. Needless to say, the calculation of that kernel has a very high computational cost. We use dynamic programming to calculate it in an efficient manner and justifiable time.

Considering the definitions of $\mathbf{I}_p^s$ and $\mathbf{I}_p^t$, we can rebuild the Eq. 5.15:

$$\mathcal{K}_p(s,t) = \sum_{i=1}^{|s|}\sum_{j=1}^{|t|} \sum_{(\mathbf{i},\mathbf{j})\in\mathbf{I}_p^{s(1:i)}\times I_p^{t(1:j)}} \kappa_p(s_{\mathbf{i}}, t_{\mathbf{j}})\gamma^{g_{\mathbf{i}}+g_{\mathbf{j}}}$$

To express the kernel using a suffix version of that, we define the suffix kernel as:

$$\mathcal{K}_p^S(s(1:i),t(1:j)) = \sum_{(\mathbf{i},\mathbf{j})\in \mathbf{I}_p^{s(1:i)}\times\mathbf{I}_p^{t(1:j)}} \kappa_p(s_{\mathbf{i}},t_{\mathbf{j}})\gamma^{g_{\mathbf{i}}+g_{\mathbf{j}}} \tag{5.16}$$

So we have:

$$\mathcal{K}_p(s,t) = \sum_{i=1}^{|s|}\sum_{j=1}^{|t|}\mathcal{K}_p^S(s(1:i),t(1:j) \tag{5.17}$$

If we add a new data point x to the time series s', using the above equation we can calculate $\mathcal{K}_p(s'x,t')$:

$$\begin{aligned}\mathcal{K}_p(s'x,t') &= \sum_{i=1}^{|s'x|}\sum_{j=1}^{|t'|}\mathcal{K}_p^S(s'x(1:i),t'(1:j)) \\ &= \sum_{i=1}^{|s'|}\sum_{j=1}^{|t'|}\mathcal{K}_p^S(s'(1:i),t'(1:j)) + \sum_{j=1}^{|t'|}\mathcal{K}_p^S(s'x,t'(1:j))\end{aligned}$$

Then,

$$\mathcal{K}_p(s'x,t') = \mathcal{K}_p(s',t') + \sum_{j=1}^{|t'|}\mathcal{K}_p^S(s'x,t'(1:j)) \tag{5.18}$$

Similar to the time–resolved spectrum, we accept a constraint on choosing the kernel function $\kappa_p(s_{\mathbf{i}},t_{\mathbf{j}})$ (Equation 5.14) and consider $\kappa^*(s_{\mathbf{i}i},t_{\mathbf{j}_i}) = \exp\frac{-(s_{\mathbf{i}_i}-t_{\mathbf{j}_i})^2}{2\sigma^2}$ to measure the similarity between two data points, then:

$$\kappa_p(s_{\mathbf{i}},t_{\mathbf{j}}) = \prod_{i=1}^{p}\kappa^*(s_{\mathbf{i}i},t_{\mathbf{j}_i}) = \exp\left(-\frac{||s_{\mathbf{i}}-t_{\mathbf{j}}||^2}{2\sigma^2}\right) \tag{5.19}$$

That, $\kappa_p(s_{\mathbf{i}},t_{\mathbf{j}})$ is a gaussian kernel of width σ and suitable for measuring the local similarity of subsequences in the time series. This also ensures the positive definiteness of our suggested kernel (Eq. 5.15).

If we add another new data point y to the time series t', considering the assumption for κ_p and the above definition of $\mathcal{K}_p^S$ (Eq. 5.17), we have:

$$\mathcal{K}_p^S(s'x,t'y) = \kappa^*(x,y)\sum_{i=1}^{|s'|}\sum_{j=1}^{|t'|}\gamma^{|s'|-i+|t'|-j}\mathcal{K}_{p-1}^S(s'(1:i),t'(1:j)) \tag{5.20}$$

It means when new points are added, to measure the new p-suffix kernel, we must calculate similarities of $p-1$ length subsequences in the suffixes considering all possible

degrees of warping. To evaluate $\mathcal{K}_p^S$ recursively, we define:

$$\mathcal{K}_p^{Sw}(k,l) = \sum_{i=1}^{k}\sum_{j=1}^{l} \gamma^{k-i+l-j}\mathcal{K}_{p-1}^{S}(s'(1:i), t'(1:j)) \tag{5.21}$$

Then equation 5.20 becomes:

$$\mathcal{K}_p^S(s'x, t'y) = \kappa^*(x,y)\mathcal{K}_p^{Sw}(|s'|, |t'|) \tag{5.22}$$

to express the above kernel recursively, we use the relation:

$$\sum_{i=1}^{a}\sum_{j=1}^{b} f(i,j) = f(a,b) + \sum_{i=1}^{a-1}\sum_{j=1}^{b} f(i,j) + \sum_{i=1}^{a}\sum_{j=1}^{b-1} f(i,j) - \sum_{i=1}^{a-1}\sum_{j=1}^{b-1} f(i,j)$$

Let $f(i,j) = \gamma^{k-i+l-j}\mathcal{K}_{p-1}^{S}(s'(1:i), t'(1:j))$, $a = k$ and $b = l$, we have the following algorithm:

Algorithm: *Recursive computation of the warped time resolved spectrum kernel.*

$$\mathcal{K}_p^{Sw}(k,l) = \mathcal{K}_{p-1}^{S}(s'(1:k), t'(1:l)) + \gamma\mathcal{K}_p^{Sw}(k,l-1) + \gamma\mathcal{K}_p^{Sw}(k-1,l) - \gamma^2\mathcal{K}_p^{Sw}(k-1,l-1)$$

$$\mathcal{K}_p^S(s'x, t'y) = \kappa^*(x,y)(x,y)\mathcal{K}_p^{Sw}(|s'|, |t'|) \tag{5.23}$$

$$\mathcal{K}_p(s'x, t') = \mathcal{K}_p(s', t') + \sum_{j=1}^{|t'|} \mathcal{K}_p^S(s'x, t'(1:j)) \tag{5.24}$$

$$\begin{aligned}
\mathcal{K}_0^S(s', t') &= 1 \quad \textbf{for} \quad \textbf{all} \quad s', t', \\
\mathcal{K}_i^S(s', t') &= 0, \quad \textbf{if} \ \min(|s'|, |t'|) < i, \\
\mathcal{K}_i(s', t') &= 0, \quad \textbf{if} \ \min(|s'|, |t'|) < i,
\end{aligned}$$

The computation of the kernel follows a dynamic programming technique with the order of $O(p|s||t|)$. We have recursions over the prefixes of the time series and the lengths of the subsequences and we do the routine above until $x = s_{|s|}$ and $|t'| = |t|$. As we see from the following pseudo-code, the evaluation of the $\mathcal{K}_i^{norm}$ is of order $O(|s||t|)$ and the overall complexity of our algorithm to calculate a linear combination of all p-spectrum kernels is $O(p|s||t|)$.

This operation scales the similarities in the range [0,1]. Fig. 5.12 plots the kernel score of two samples of echoes reflected by a Ficus tree with different values of warping cost. We see that as the gamma parameter gets closer to 1 we let subsequences of two time series warp more and the similarity score (kernel score) increases. When gamma is equal to zero, the kernel is equal to the time-resolved spectrum kernel.

Algorithm Warped Time resolved spectrum kernel

Input : Time series s and t of length n and m, max subsequence length l and warping cost γ;

Output: Array of spectrum kernel $\mathcal{K}[]$ with different sizes of subsequence-length from 1 to l;

$KPSw(0:n, 0:m) = 0$;
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** m **do**
    $KPS(i,j) = \kappa^*(s_i, t_j)$;
    $\mathcal{K}[1] = \mathcal{K}[1] + KPS(i,j)$;
  end
end
for $p \leftarrow 2$ **to** l **do**
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** m **do**
      $KPSw(i,j) = KPS(i-1,j-1) + \gamma KPSw(i,j-1) +$
      $\gamma KPSw(i-1,j) - \gamma^2 KPSw(i-1,j-1)$;
      $KPS(i,j) = \kappa^*(s_i, t_j) KPSw(i-1,j-1)$;
      /* Equation 5.23 */
      $\mathcal{K}[p] = \mathcal{K}[p] + KPS(i,j)$;
      /* Equation 5.24 */
    end
  end
end
return $\mathcal{K}[]$

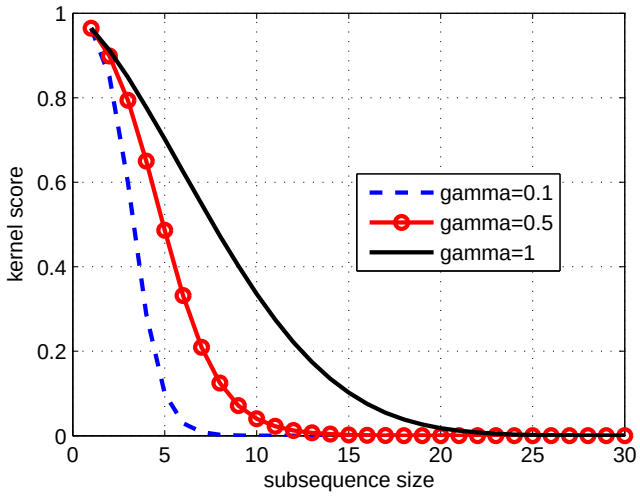


Figure 5.12: Kernel score between two echoes of Ficus tree with different warping costs

5.6 Classification and Results

We gathered the sonar data,720 echoes for each tree (Figure 5.4), as explained before. After the preprocessing steps for each echo (Fig. 5.5), we have a time series in which each point is a time frame and its value is an array of features (the average energy of each channel of gammatone filter). Using the spectrum kernel method told in the previous section, we want to extract the similarities between the echoes for our classification task.

5.6.1 Time–resolved spectrum kernel

As we told in the above, the time–resolved kernel is a special case of the warped time-resolved spectrum kernel which the warping cost $\gamma = 0$. According to Equation 5.12 and 5.13, we need to find the parameters θ_i and σ.

Finding the parameters θ_i is a case of more general problem known as optimal kernel selection. In the next chapter we will discuss our method for selection of the optimal kernels. However, here for simplicity, we consider equal values of θ_i in the range $[p_1, p_2]$ as follows:

$$\theta_i = \begin{cases} 1 & p_1 \leq i \leq p_2 \\ 0 & \text{otherwise} \end{cases}$$

p_1 and p_2 are the minimum and maximum sizes of subsequences used to extract the similarities in each tree. To find suitable values for those parameters, we used a simple *grid search* on p_1 and p_2. We selected randomly 100 echoes of each tree and then calculated $\mathcal{K}_i^{norm}(s[m], s[n])$ for $i \in [1, l]$, $m, n \in [1, 100]$ and $\sigma \in \{1, 10, 100, 1000\}$ where $s[m]$ and $s[n]$ are the m-th and n-th of pre-processed echoes and l is the length of the time

series (in our experiment 90). Then we found the optimum values p_1 and p_2 in the range [1,l] for each σ, by maximizing the average value of the kernels:

$$\max_{p_1,p_2} K = \sum_{m=1}^{100} \sum_{n=1}^{100} \left(\frac{\sum_{i=p_1}^{p_2} \mathcal{K}_i^{norm}(s[m], s[n])}{p_2 - p_1} \right)$$

We found that a suitable value for σ is in the range [10,100] for all trees. Table 5.6.1 shows the optimum values for p_1 and p_2 with $\sigma = 10$. We see that for ficus and bamboo, which have smaller leaves the p_1 has lower value. Again, for simplicity, we considered equal values of p_1 and p_2 for all trees in our classification method.

Tree	p_1	p_2
Ficus	5	25
Bamboo	8	23
Schefflera	11	32

Table 5.3: Optimum Values for p_1 and p_2.

To measure the robustness of our algorithm, we randomly selected 100 echoes of each tree (total 300 echoes) to train the classifier. Considering the same σ, p_1 and p_2 ($\sigma = 10$, p_1=5, p_2=30) for all trees, we calculated the kernel matrix $\mathbf{K}$:

$$\mathbf{K}(i,j) = K(s[i], s[j]) = \sum_{l=p_1}^{p_2} \mathcal{K}_l^{norm}(s[i], s[j])$$

in which i, $j \in [1, 300]$ and $s[i]$ is i-th echo, where for Ficus echoes, $i \in$[1,100], for Bamboo $i \in$ [101,200] and for Schefflera $i \in$[201,300].

After calculation of the kernel matrix $\mathbf{K}$, we used the LIBSVM package for Support Vector Machines (SVMs) regression and classification [31]. It lets us use our own kernel matrix to train the classifier. We used the remaining data (1860 echoes) for test.

The prediction quality was then evaluated by specificity (Spec.), sensitivity (Sen.) and accuracy (Acc.) as follows:

$$\begin{aligned} Sen. &= \frac{TP}{(TP+FN)} \\ Spec. &= \frac{TN}{(TN+FP)} \\ Acc. &= \frac{TP+TN}{(TN+FN+TP+FP)} \end{aligned} \tag{5.25}$$

where TP = number of true positives, TN = number of true negatives, FP = number of false positives and FN = number of false negatives.

Table 5.6.1 shows the average performance of the classifier. It should be noted that the classifier decides based on only one observation. If we use more observations and decide based on the average of the probability that an observation belongs to a class, the accuracy increases (discussed in the next chapter).

Tree	Specificity (%)	Sensitivity (%)	Accuracy (%)
Ficus	85.2	87.5	86.3
Bamboo	87.1	90.1	89.3
Schefflera	92.8	94.5	93.8

Table 5.4: Performance of time-resolved spectrum kernel in biosonar landmarks classification with 100 randomly selected echoes for training.

5.6.2 Warped time–resolved spectrum kernel

We find the parameters p_1 and p_2 with different values of warping cost γ in the warped time–resolved spectrum kernel and then calculate the kernel matrix. Table 5.6.2 shows the performance of the classifier for different values of γ when it decides based on only one observation.

We see that by changing the parameter γ the accuracy of classifier changes. The best accuracy for Ficus, Bamboo and Schefflera trees are gained with $\gamma = 0.1$, $\gamma = 0.3$ and $\gamma = 0.2$, respectively. This parameter lets the kernel consider a warping (with a cost) for the subsequences of the time series and extract their similarity. Considering that parameter in our classification task is justifiable, because the echoes reflected by the adjacent leaves of each tree can have somehow similar patterns but not exactly the same, so we need to have a parameter (γ) that can let the kernel capture those similarities, too. The optimal value of that parameter for each tree can be related to the physical specification of each tree.

Tree	Specificity (%)	Sensitivity (%)	Accuracy (%)
Ficus (γ=**0.1**)	**86.3**	**88.2**	**87.1**
Ficus (γ=0.2)	85.1	82.1	84.3
Ficus (γ=0.3)	81.1	83.4	82.7
Bamboo (γ=0.1)	85.9	87.8	86.6
Bamboo (γ=0.2)	86.1	91.1	89.1
Bamboo (γ=**0.3**)	**88.6**	**90.4**	**89.8**
Schefflera(γ=0.1)	91.7	89.1	90.6
Schefflera(γ=**0.2**)	**92.8**	**94.4**	**93.7**
Schefflera(γ=0.3)	92.6	94.1	93.1

Table 5.5: Performance of warped time-resolved spectrum kernel in biosonar landmarks classification with 100 randomly selected echoes for training.

5.7 Conclusion

We considered the problem of biosonar landmark classification as an example of random and non stationary signal classification in which finding robust features for classification is not trivial. We regarded both the local temporal similarity and the power spectrum of echoes and suggested a kernel based classification method that extracts those local similarities, independent of the position of occurrences in echoes of each object. We suggested a kernel called *time-resolved spectrum kernel* to measure the similarity of echoes as time series and made a relation between that kernel and geometric specification of the objects. The p-length subsequence of that kernel simply measures the occurrences of fixed p-length subsequences for each of the time series in consideration. The more time series share similar p-length subsequences, the more similar they are.

We also proposed a more general kernel called *warped time-resolved spectrum kernel*, which considers warping in the subsequences. We then used those kernels directly in a SVM-based classifier. We think this kind of kernel is suitable for pattern recognition in signals with inherent self similarity and for estimating periodicity in arbitrary time series like speech and biomedical signals. In our method, to keep the problem simple, we made a not very accurate assumption for the θ_i parameters of the kernel with an equal value for all θ_i. But that parameter, the weight of the similarity (kernel score) of the p-size subsequences, can represent the self similarity of one part of the object, for example the size of leaves, and also can show the geometric characteristics of that object. In the next chapter, we will try to find the optimum value of those parameters while maximizing the accuracy of the classifier using an optimization algorithm.

Chapter 6

Kernel Selection in Time series Kernels

6.1 Introduction

In the previous chapter, we considered a class of random process signals and time–series which contain sub similarities at random positions representing the texture of an object. Those repetitive parts may occur in speech, musical pieces and sonar signals. We suggested a time–resolved spectrum kernel for extracting the subsequence similarity in time series in general, and as an example in biosonar signals. In our classification task, we considered the similarity of subsequences having different sizes and a linear combination of different i-spectrum kernels with different weighting $\theta_i \geq 0$:

$$K(s,t) = \sum_{i=1}^{l} \theta_i \mathcal{K}_i^{norm}(s,t) \tag{6.1}$$

The parameters θ_i show the weight of each i-length kernel and the optimum selection of those parameters extracts maximum similarities in the signals in consideration. We then proposed a non–optimal and simple method to find a combination of those kernels.

The kernel selection problem has been studied using different methods. Fung et al. [47] formulated an optimal kernel selection based on the quadratic programming formulation of the Fisher linear discriminant. They developed an iterative method that alternates between optimizing the weight vector and the Gram matrix. Kim et al. [108] considered the kernel selection in terms of maximization of the Fisher discriminant ratio and showed that the kernel selection can be formulated as a tractable convex optimization problem, and hence the globally optimal kernel can be found with efficiency.

The kernel Fisher discriminant analysis is a non-linear extension of the linear Fisher discriminant analysis. It finds the direction in a feature space, defined implicitly by a kernel, onto which the projections of positive and negative classes are well separated in terms of the Fisher discriminant ratio. This criterion ascertains that the obtained kernel maximizes the similarity score between signals of one class and minimizes the similarity score between signals of two different classes.

In this chapter, we find the optimal kernel via maximizing the Kernel Fisher Discriminant criterion (KFD) [92] to build the optimal linear combination of kernels. Given that criterion, to solve the optimization problem, unlike to the work of Kim et al. [108] we use a faster search method called Mesh Adaptive Direct Search (MADS). MADS as defined by Audit and Dennis [6] is a class of algorithms for nonlinear optimization. It computes a series of points that get closer and closer to the optimal point. The algorithm searches a set of randomly selected points, called a mesh, around the current point–the point computed at the previous step of the algorithm. The mesh is formed by adding the current point to a scalar multiple of a set of vectors called a pattern and the point in the mesh that improves the objective function becomes the current point at the next step. The routine continues until a stopping criterion is fulfilled.

We use this method [13, 18] to find the optimum value of the parameter θ_i (equation 6.1) as an optimum weight of the kernel in the kernel selection problem. Using the obtained kernel, we then use the SVMs classifier to classify different classes of biosonar signals. We get more accurate results compared with the results of the previous chapter, where we used a simple method for the kernel selection.

This chapter organized as follows: section 6.2 contains the optimal kernel selection using the Fisher discriminant criterion. In section 6.3, the mesh adaptive direction search method is described. The experimental results are presented in section 6.4, and section 6.5 draws the conclusion of the work.

6.2 Fisher Discriminant based Optimal Kernel Selection

Having two classes of labelled data, Fisher's idea was to look for a direction v that separates the class means well (when projected onto the found direction) while achieving a small variance around these means. The hope is that, using this projection a classifier can classify the unlabelled data with a small error. The quantity measuring the difference between the means is called between class variance and the quantity measuring the variance around these class means is called within class variance, respectively. Then, in linear Fisher discriminant analysis, the goal is to find a direction that maximizes the inter–class variance while minimizing the intra–class variance at the same time (Fig. 6.1).

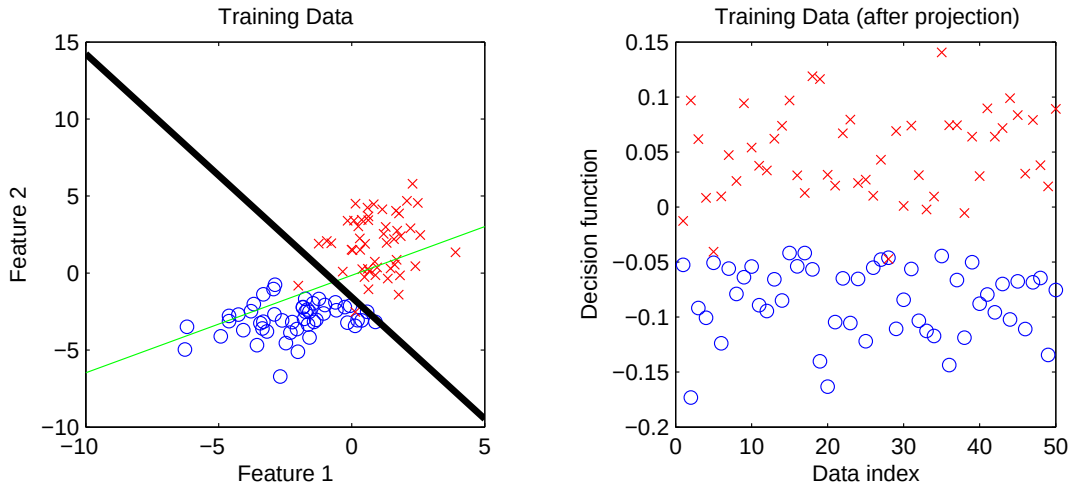
Given a set of n_+ positive training data $\chi_+ \subset \mathbb{R}^d$ (positive class), a set of n_- negative data $\chi_- \subset \mathbb{R}^d$ ($\chi = \chi_+ \cup \chi_-$, all data) and binary labels $y_i \in \{-1, 1\}$ indicating the two classes, the class separability in a direction $v \in \mathbb{R}^d$ is defined as:

$$J(v) = \frac{< v.S_B v >}{< v.S_W v >} \tag{6.2}$$

where S_B is the inter–class scatter matrix

$$S_B = (m_+ - m_-)(m_+ - m_-)^T$$

in which



(a) Projection (b) Linear discriminant function

Figure 6.1: Linear Fisher projection. a) Projection axis (green) and decision boundary between means of both classes (black bold). b) Linear discriminant function

$$m_+ = \frac{1}{n_+} \sum_{x \in \chi_+} x, \; m_- = \frac{1}{n_-} \sum_{x \in \chi_-} x,$$

and S_W is the intra–class scatter matrix defined as:

$$S_W = S_+ + S_-$$

where $S_- = \sum_{x \in \chi} (x - m_-)(x - m_-)^T$ and $S_+ = \sum_{x \in \chi} (x - m_+)(x - m_+)^T$.

In signal processing, this criterion (class separability) is known as the signal-to-interference ratio.

In the case of the Fisher Linear Discriminant (FLD), the parameter vector v of the linear discriminant function $f(x) = \text{sgn}(< v.x > + b)$ is determined to maximize the class separability (signal-to-interference ratio). Then the problem is to find:

$$v_{opt} = \arg\max_v \; J(v) = \arg\max_v \frac{< v.S_B v >}{< v.S_W v >} \tag{6.3}$$

The classical solution from linear algebra to the above problem is:

$$v_{opt} = S_W^{-1}(m_+ - m_-) \tag{6.4}$$

It can also be solved using quadratic programming. Fig. 6.1 shows an example of the linear Fisher projection and its decision function on a sample data set.

Linear discriminants are not always optimal for classification, especially in nonlinear feature space, where we need nonlinear decision functions. Using the kernel trick, the

KFDA first maps the data via a non-linear mapping ϕ into the high dimensional feature space $\mathcal{F}$ and then optimizes the Fisher criterion.

Given a map $\phi : \ u \rightarrow \phi(u) \in \mathcal{F}$, the aim is to find a direction $v = \sum_{i=1}^{n} \alpha_i \phi(u_i)$ in the feature space $\mathcal{F}$ given by weights $\alpha = [\alpha_1, ..., \alpha_n]$, that maximizes the separation of the mean scaled in the feature space and minimizes the variance in that direction (KFD criterion). Considering the kernel matrix K:

$$K_{i,j} = k(x_i, x_j) =< \phi(x_i), \phi(x_j) > \tag{6.5}$$

For the direction v, the criterion shown in Eq. 6.2 will be in the form of ([92]):

$$J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T (N + \lambda I) \alpha} \tag{6.6}$$

The parameter λ is a regulation factor and M and N (defined in [92]) are gained in terms of the kernel matrix K:

$$M = (\mu_+ - \mu_-)(\mu_+ - \mu_-)^T$$

where $\mu_+ = \frac{1}{n_+} \sum_{x \in \chi_+} K_x$, and $\mu_- = \frac{1}{n_+} \sum_{x \in \chi_-}$ are scaled means in the feature space, and:

$$N = KDK^T \tag{6.7}$$

where

$$D = \begin{bmatrix} I_{n_+} - \frac{1}{n_+} 1_{n_+} 1_{n_+}^T & 0 \\ 0 & I_{n_-} - \frac{1}{n_-} 1_{n_-} 1_{n_-}^T \end{bmatrix}_{n \times n}$$

in which 1_n and I_n denote the vector of all ones and the identity operator in $\mathbb{R}^d$, respectively.

Comparing to the solution of the linear Fisher discriminant (Eq. 6.4), the parameter α that maximizes Eq. 6.6 is obtained via:

$$\alpha_{\max} = (N + \lambda I)^{-1} (\mu_+ - \mu_-) = (KDK^T + \lambda I)^{-1} Ky \tag{6.8}$$

where:

$$y = \begin{bmatrix} (1/n_+) 1_{n_+} \\ (-1/n_-) 1_{n_-} \end{bmatrix}_{n \times 1}$$

which results in:

$$J_{\max}(K) = \alpha_{\max}^T Ky = y^T K (KD^T K + \lambda I)^{-1} Ky \tag{6.9}$$

If we consider the variable K as a linear combination of a set of kernel matrices, in the next step we try to find the matrix K, which maximizes the above equation. Considering equations 6.1 and 6.9, the problem of finding the optimal kernel in terms of maximizing

the Fisher discriminant ratio can be written as:

$$\begin{array}{cc} \min & f(\sum_{i=1}^{l}\theta_i\mathcal{K}_i^{norm}) = -J_{\max}(\sum_{i=1}^{l}\theta_i\mathcal{K}_i^{norm}) \\ \text{subject to} & \theta \geq 0, \quad 1^T\theta = 1 \end{array}$$

It is easy to prove the convexity of the above objective function. We suppose $f(x,y) = x'y^{-1}x$, $h(K) = Ky$ and $g(K) = KD'K + \lambda I$, considering the convexity of f, h and g, we conclude the convexity of $f(h,g)$ and so the above objective function. Then, any local optimum answer for the objective function is a global one of that, too. One suggested method (Kim et. al [108]) was to use the convex optimization and bring the objective function in the form of Semi-Definite Programming (SDP) via the Schur complement technique [26]. In their method, the SDP solver of SeDuMi [121] was used to solve the SDP.

Instead of that method we use a newly suggested method for local optimization known as *Mesh Adaptive Direct Search* method that needs less run time (approximately one third in our experiments) on a PC with an Intel Core Duo processor (1.83GHz and 1GB RAM) while coded in Matlab 7.0.

6.3 Mesh Adaptive Direct Search

MADS (defined by Audit and Dennis [6]) is a class of algorithms for nonlinear optimization. It is a modification of the generalized pattern search (GPS [5]) algorithm for local optimization. In summary, this algorithm computes a series of points that get closer and closer to the optimal point. It searches a set of randomly selected points, called a *mesh*, around the *current point*–the point computed at the previous step of the algorithm. The mesh is formed by adding the current point to a scalar multiple of a set of vectors called the *mesh size* (pattern). (The GPS algorithm uses fixed direction vectors, whereas the MADS algorithm uses a random selection of vectors to define the mesh). The point in the mesh that improves the objective function becomes the current point at the next step. The value of the objective function either decreases or remains the same from each current point to the next. The routine continues until a stopping criterion is fulfilled. The formal definitions and algorithm from [6] follow:

Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}$ is a given function under general constraint $x \in \Omega \subseteq \mathbb{R}^n$, $\Omega \neq 0$. If $\Omega \neq \mathbb{R}^n$ (constraint optimization), the algorithm attempts to locate a minimizer of function f over Ω by means of a *barrier* function:

$$f_\Omega = \begin{cases} +\infty & \text{if } p \notin \Omega \\ f & \text{otherwise.} \end{cases}$$

The algorithm does not require the use of the approximations or derivatives of f (free-derivative method). This is useful (especially in our case) when ∇f is not available or can not be accurately estimated.

MADS is an iterative algorithm where at each iteration k a finite number of trial points are generated and their objective function values are compared with the current incumbent value $f_\Omega(x)$ as the best objective function value found so far. Each of these trial points lies on the *current mesh*, constructed from a finite fixed set of n_D directions $D \subset \mathbb{R}^n$ scaled by a *mesh size* parameter $\Delta_k^m \in \mathbb{R}_+$. The mesh size parameter controls the coarseness or fineness of search at iteration k. Δ_{k+1}^m is adjusted from Δ_k^m depending on the success of that iteration.

$D_{n\times n_D}$ must be a positive spanning set, *i.e.*, nonnegative linear combinations of its elements must span $\mathbb{R}^n$, and each direction $d_j \in D$ ($j \in [1, n_D]$) must be the product of some fixed nonsingular generating matrix $G \in \mathbb{R}$ by an integer vector $z_j \in \mathbb{Z}^n$. We consider Z a matrix whose columns are z_j, for $j = 1, 2, ..., n_D$, and use matrix multiplication, $D = GZ$. If S_k is the set of points where the objective function f had been evaluated at the start of iteration k, at iteration k, the current mesh is defined as:

$$M_k = \bigcup_{x\in S_k} \{x + \Delta_k^m Dz : \ z \in N^{n_D}\}$$

The above definition ensures that all previously visited points lie on the mesh, and that new trial points can be selected around any of them.

Each iteration consists of SEARCH and POLL steps. In the SEARCH step the value of f_Ω at any finite number of mesh points is evaluated. When a improved mesh point, at which f_Ω is less that $\min_{x\in S_k} f_\Omega$, is generated, the iteration may stop, or it may continue to find a better improved mesh point. Otherwise the POLL steps begins and the algorithm generates and evaluates f_Ω around the current incumbent x_k, where $f_\Omega(x_k) = \min_{x\in S_k} f_\Omega(x)$. The poll size parameter Δ_k^p limits the distance between x_k and the new trail points. The set of new trail points is called a *frame* and x_k is the *frame center*. This frame is generated using x_k, Δ_k^p, Δ_k^m, and D to obtain a set D_k of positive spanning directions.

At iteration k, the *MADS frame* is defined to be the set:

$$P_k = \{x_k + \Delta_k^m d : \ d \in \ D_k\} \subset M_k$$

in which D_k is a positive spanning set ($0 \notin D_k$) and for each $d \in D_k$:

- $d \neq 0$ is nonnegative integer combination of the directions in D,
- $\Delta_k^m d$, the distance from the frame center, is bounded by a constant times the poll size parameter: $\Delta_k^m \|d\| \leq \Delta_k^p \ \max\left\{\left\|d'\right\| : d' \in D\right\}$,
- limits of the normalized set D_k are positive spanning sets [33].

To ensure the convergence, the radii of successive frames must converge to zero at a slower rate than the mesh size parameter. It means $\Delta_{k+1}^m \leq \Delta_{k+1}^p$ and it must satisfy

$$\lim_{k\to\infty} \inf \Delta_k^p = \lim_{k\to\infty} \inf \Delta_k^m = 0$$

The algorithm evaluates f_Ω at points in the frame P_k until it finds an improved point t with $f_\Omega(t) < f_\Omega(x_k)$ or until it has evaluated f_Ω at all of the points in P_k. When the POLL step fails to generate an improved mesh point then the frame is called a *minimal frame*, and the fame center p_k is said to be a minimal frame center and the poll size parameter should be updated.

At iteration k, the mesh size parameter is updated according to:

$$\Delta_{k+1}^m = \begin{cases} \Delta_k^m/4, & \text{if } p_k \text{ is a minimal frame center} \\ 4\Delta_k^m, & \text{if an improved mesh point is found, and } \Delta_k^m \leq \frac{1}{4} \\ \Delta_k^m, & \text{Otherwise.} \end{cases}$$

and the poll size parameter as:

$$\Delta_k^p = \sqrt{\Delta_k^m}$$

These rules guarantee that Δ_k^m is always a power of 1/4 and less than or equal to one, and $\Delta_k^m \leq \Delta_k^p$ for all k. We can select a default minimum value of mesh size as stopping criterion to be fulfilled.

In summary, the MADS algorithm is described as follows:

Algorithm Mesh Adaptive Direct Search Algorithm

---

step 0 [Initialization] Given x_0, $\Delta_k^m \leq \Delta_k^p$ and $D_{n\times n_D}$ a positive spanning set. Set $\Delta_0^m = 1$, and the iteration counter k:=0

step 1 [SEARCH step] Evaluate f_Ω on a finite subset of trial points on the mesh M_k as defined above. If an improved trail point t with $f_\Omega(t) < f_\Omega(x_k)$ is found, declare k successful and go to step 3.

step 2 [POLL step] Evaluate f_Ω at points from the frame P_k until an x_{k+1} with $f_\Omega(x_{k+1}) < f_\Omega(x_k)$ is found. If no such point exists, declare k unsuccessful.

step 3 [Parameter update] If iteration k was declared unsuccessful, then set $p_{k+1} = p_k$ (minimal frame center). Otherwise p_{k+1} is an improved mesh point. Update Δ_{k+1}^m and Δ_{k+1}^p. If an appropriate stopping criteria has not been met, set k:=k+1 and go to step 1.

6.4 Experiment and Results

Similar to the experiment explained in the previous chapter, we gathered the sonar data, 720 echoes for each tree. After the preprocessing steps for each echo (Fig. 5.5), we selected randomly 100 echoes of each tree and then calculated $\mathcal{K}_i^{norm}(s[m], s[n])$ for

$i \in [1, l],\ m, n \in [1, 100]$, where $s[m]$ and $s[n]$ are the m-th and n-th of pre-processed echoes and l is the length of the windowed time series (here 90). Using the optimal kernel selection noted above, we found the optimal value for θ_i in Eq. 6.1 and calculated the matrix K:

$$\mathbf{K}(i, j) = K(s[i], s[j]) = \sum_{k=1}^{l} \theta_i^{opt} \mathcal{K}_l^{norm}(s[i], s[j]) \tag{6.10}$$

in which $i, j \in [1, 300]$ and $s[i]$ is i-th echo. For Ficus echoes $i \in$[1,100], for Bamboo $i \in$ [101,200] and for Schefflera $i \in$[201,300]. For that we needed to fix parameters σ and γ. In our experiments, we found the suitable value of σ (Eq. 5.12) in the range [10,100].

Figure 6.2 shows the accuracy of the classifier for those trees with the optimum parameters of θ_i, different warping costs (γ) and $\sigma = 100$, based on the number of echoes as observation. It shows a high accuracy even for a low number of echoes. We see that the parameter γ can affect the accuracy of the classifier and the accuracy of the time-resolved spectrum kernel (γ=0, without warping) increases in each tree by changing the parameter γ.

Table 1 shows the accuracy of the classifier when it decides based on only one observation. The best accuracies for Ficus, Bamboo and Schefflera trees are gained for $\gamma = 0.1$, $\gamma = 0.3$ and $\gamma = 0.2$, respectively. We see if γ gets closer to 1 (no cost for warping) the accuracy decreases. Comparing the above results with the results in the previous chapter (table 5.6.2), in which we used a simple method for kernel selection, it shows an improvement in the accuracy of classification.

γ	Ficus	Bamboo	Schefflera
$\gamma = 0.0$	86.2	89.5	90.2
$\gamma = 0.1$	**89.1**	90.1	91.3
$\gamma = 0.2$	88.6	91.4	**93.8**
$\gamma = 0.3$	87.6	**93.1**	91,1
$\gamma = 0.5$	80.1	81.3	82.1
$\gamma = 1.0$	59.2	67.4	58.1

Table 6.1: Classification rate based on different values of γ and optimum kernel selection

Comparing with the previous works of our group (Wang et al. [135, 134]), the new classifier shows a notable improvement in accuracy. The best result for classification gained before was through template matching in 2D biosonar acoustic images (using a 2D Discrete Cosine Transform). The classification was made via extracting the maximum normalized cross correlation between the acoustic templates (Fig. 6.4). As shown in Fig. 6.3, we could get higher accuracy in both single and repeated observations (even with fewer echoes) compared with Fig. 6.4 (note the different horizontal and vertical axes).

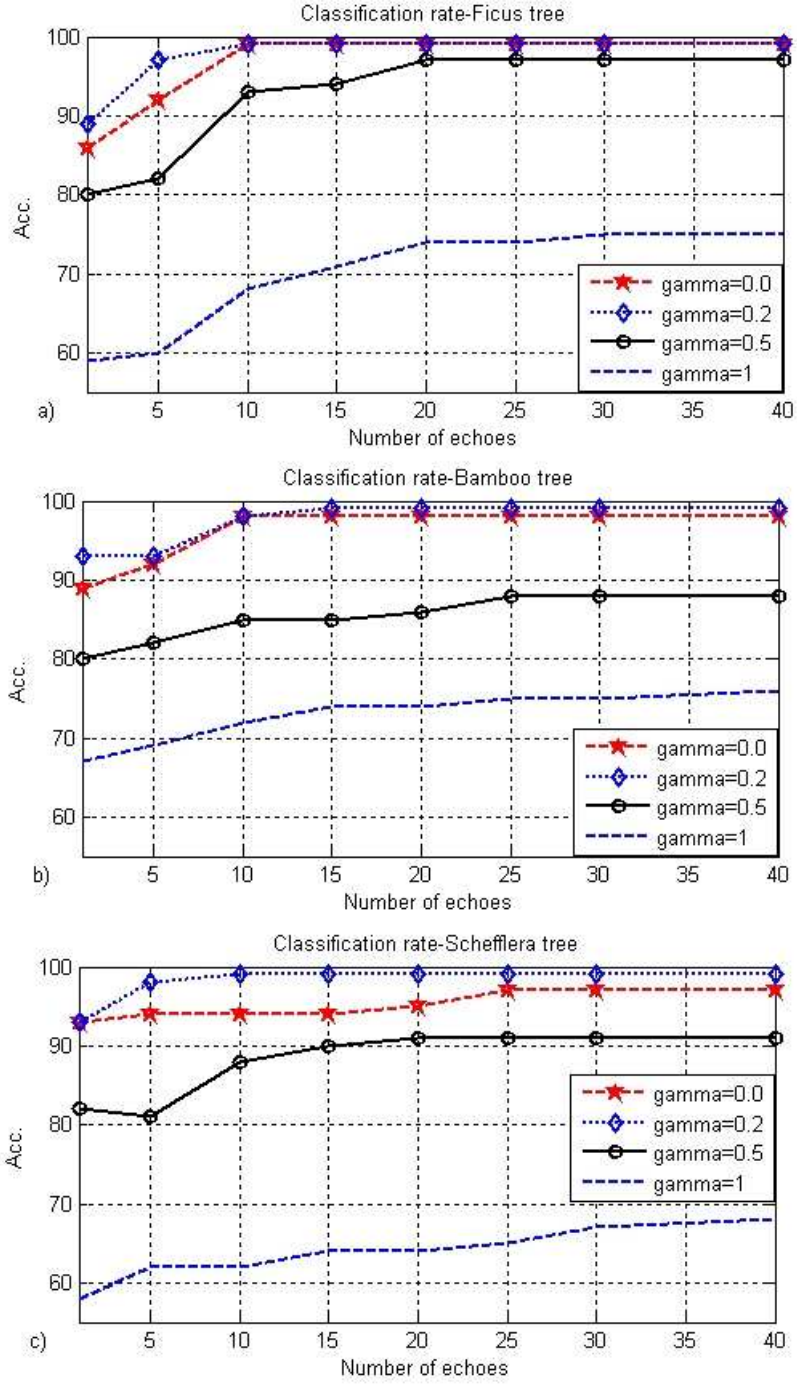


Figure 6.2: The accuracy of the classifier using our suggested kernel with different warping costs ($\sigma = 100$) for a) Ficus, b) Bamboo and c) Schefflera. For $\gamma = 0$, the kernel is similar to the time-resolved spectrum kernel [10].

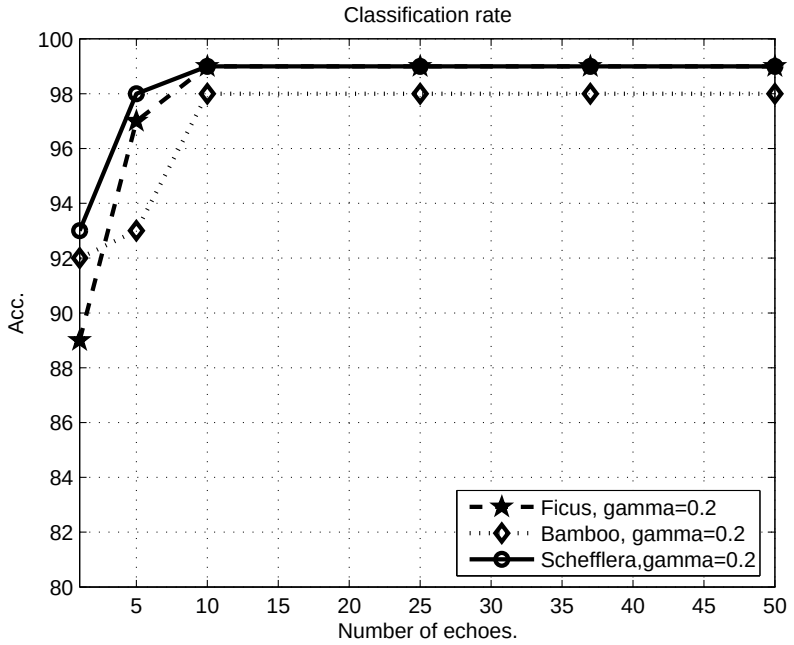


Figure 6.3: The accuracy of classifiers using different numbers of echoes for testing with the Warped time-resolved spectrum kernel ($\gamma = 0.2$).

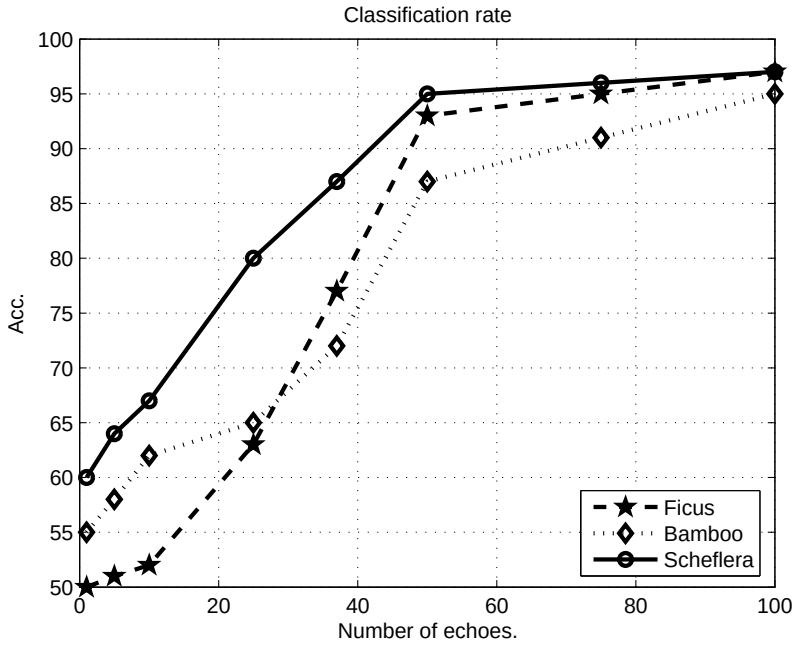


Figure 6.4: The accuracy of classifiers via template matching using acoustic images of echoes (Wang et al. [134]).

6.5 Conclusion

In this chapter, we presented a method to extract an optimal linear set of time-resolved spectrum kernel based on the Fisher criterion in kernel space. With this criterion, the obtained kernel ensures the maximum similarity score of signals of one object and the minimum similarity score between signals of two different objects. We used a Mesh Adaptive Direct Search method (MADS) to solve our optimization problem. This optimization method needs less run time than another suggested method [108] that brings the objective function in the form of Semi-Definite Programming (SDP) via the Schur complement technique. Compared with other matching methods for biosonar signals [135, 134], we obtained better results with our suggested kernel based method. Despite the accurate rate of classification, the main drawback of the time-resolved spectrum kernel is the low speed of both training and testing procedures. The problem is more acute when we want to optimize the kernel selection. It prevents us to use the method for real-time applications, although this was not our main aim. In the next chapter we will see how the boosting method, when it is used with the kernel functions, can resolve this problem.

Chapter 7

A Kernel Based Boosting method for Biosonar Data classification

7.1 Introduction

In chapters 5 and 6 we suggested a kernel named *Time-resolved spectrum kernel* for matching the subsequences of time series (sonar echoes) and extracting the local similarities of echoes. The results outperformed other matching techniques [135, 134]. The time-resolved spectrum kernel simply measures the whole similarities of all subsequences of the time series in consideration. The more two time series share similar subsequences, the more similar they are. An optimal linear combination of kernels with different subsequence size (p-spectrum kernels) was a measure of similarity between two time series (chapter 6). Despite the accurate rate of classification, both training and testing were slow and the method was not applicable for real applications.

In this chapter, we implement a simple, yet powerful, method for the problem at hand using gradient boosting [16]. Gradient boosting is a machine learning approach, that builds one strong classifier from many base learners. Originally, booting has been proposed in the 90's (Freund and Schapire, 1996 [45]) as a method for classification and regression in which a fitting method or estimator, called the base learner, is fitted multiple times on re-weighted data and the final boosting estimator is then constructed via a linear combination of those base learners. In different works, it has been shown that boosting outperformed other machine learning methods for high-dimensional data. It is empirically illustrated in Bühlmann and Yu [28] that boosting has mainly an advantage for data with high-dimensional predictors. Hoffmann et al. [61] used gradient boosting to classify high dimensional EEG signals in brain-computer interfaces and Jiao et al. [70] used this method for high dimensional protein classification and obtained satisfying results.

Similar to the above researches, we are also facing high-dimensional data in classification of sonar signals reflected by different kinds of trees. After the preprocessing steps for each echo (Fig. 5.5), we had a matrix of time series in which each cell was a time frame and its value was the average energy of each channel of gammatone filter. Furthermore,

we had:

$$A = C \times S \tag{7.1}$$

where A was the number of features, with C the number of channels and S the number of samples in each channel. Now, the problem is very similar to a multichannel EEG classification in which the state of brain or a special event is recognized and classified using a multichannel recording of EEG signals. So we can use the gradient boosting algorithm for the sonar classification task as Hofmann et al. [61] did for the event classification from EEG signals.

In this chapter, we study the efficiency of boosting methods for our classification task. We use the gradient boosting method with two kinds of base learners. The first one uses Ordinary Least Squares (OLS) regression and the other one uses the kernel function as base learner.

Compared with our previous works in chapters (5 and 6), in which we presented a time resolved spectrum kernel to extract the similarities between echoes, we get more efficient and accurate results with the newly proposed boosting method. We compare the methods in terms of sensitivity, specificity, accuracy and Matthew's correlation coefficient and also the runtime of training and testing.

The content of the rest of this chapter is as follows: In the following section, we describe the gradient boosting in details. In section 7.3, we present two base learners for gradient boosting. The experimental results are presented in section 7.4, and section 7.5 draws the conclusion of this work.

7.2 Gradient Boosting Algorithm

We here give a summary of the gradient boosting algorithm from [46]. Given a set of random input variables $x = \{x_1, ..., x_n\}$ and a random output variable y and some samples $\{y_i, x_i\}_{i=1}^N$, we want to find an approximation function F^* that can predict y from x such that over the joint distribution of y, x values, the expected value of a specific loss function $l(y, F(x))$ is minimized:

$$\begin{aligned} F^*(x) &= \arg\min_{F(x)} E_{y,x} l(y, F(x)) \\ &= \arg\min_{F(x)} E_x[E_y l(y, F(x))|x] \end{aligned} \tag{7.2}$$

Examples of different loss functions include squared error $(y - F)^2$ and absolute error $|y - F|$ for regression, and negative binomial log-likelihood, $\log(1 + e^{-2yF})$, when $y \in \{-1, 1\}$ for classification.

$F(x)$ is a member of parameterized class of functions $F(x; \mathbf{p})$, where $\mathbf{p} = \{p_1, p_2, ...\}$ is a finite set of parameters whose joint values identify individual class members. In the gradient boosting method, we have the additive expansions of the form

$$F(x;\{\beta_m,\mathrm{a}_m\}_{m=1}^{M})=\sum_{m=1}^{M}\beta_m h(x;\mathrm{a}_m) \tag{7.3}$$

in which $\mathbf{p}=\{\beta_{\mathbf{m}},\mathrm{a}_{\mathbf{m}}\}$. The generic function $h(x;\mathrm{a})$ is called a base or weak learner and is a simple function of x with parameters $\mathrm{a}=\{\mathrm{a}_1,\mathrm{a}_2,...,\mathrm{a}_{\mathrm{M}}\}$. The task is to find the parameters of weak learners through solving Eq. 7.2. For that, a typical parameter optimization method "greedy-stagewise" is used in which we optimize $\{\beta_m,\mathrm{a}_m\}$ after all of the $\{\beta_i,\mathrm{a_i}\}(\mathrm{i}=1,...,\mathrm{m}-1)\}$ are optimized. Then, the recursive representation of the optimization method is as follows:

$$\{\beta_m,\mathrm{a_m}\}=\arg\min_{\beta,\mathrm{a}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\mathrm{l}(\mathrm{y_i},\mathrm{F_{m-1}(x_i)}+\beta\mathrm{h}(\mathrm{x_i};\mathrm{a})) \tag{7.4}$$

where the joint distribution of (x,y) is estimated by a finite data sample $\{y_i,x_i\}_1^N$, and we have

$$F_m=F_{m-1}+\beta_m h(x;\mathrm{a}_m) \tag{7.5}$$

$\beta_m h(x;\mathrm{a}_m)$ is an incremental *boost* and the best greedy direction step towards the data-based estimate of $F^*(x)$. Friedman [46] suggested a steepest-descent method to find that direction:

$$-g_m(x_i)=-\left[\frac{\partial l(y_i,F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \tag{7.6}$$

It gives the best steepest-descent step direction at F_{m-1}. We find the parameters $\mathrm{a_m}$ that produces $\mathrm{h}_m=\{h(x_i;\mathrm{a_m})\}_{\mathrm{i}=1}^{\mathrm{N}}$ most parallel to $-g_m\in\mathbb{R}^N$. So we have:

$$\mathrm{a_m}=\arg\min_{\mathrm{a},\rho}\sum_{i=1}^{N}\left[-g_m(x_i)-\rho h(x_i;\mathrm{a})\right]^2 \tag{7.7}$$

and then Eq. 7.4 is converted to:

$$\beta_m=\arg\min_{\beta}\sum_{i=1}^{N}l(y_i,F_{m-1}(x_i)+\beta h(x_i;\mathrm{a}_m)) \tag{7.8}$$

We consider a regularization term to avoid the resulting overfit problem by a large number of weak learners. This can be done by adding a *shrinkage factor* $0<\nu\leq 1$ to the Eq. 7.5:

$$F_m=F_{m-1}+\nu\beta_m h(x;\mathrm{a}_m) \tag{7.9}$$

This can greatly improve the generalization performance of the algorithm. The general framework of the gradient boosting is as follows:

Algorithm A general gradient boosting framework

$F_0(x_i) = 0,\ \forall i$;
for $m \leftarrow 1$**to** M **do**
  $g_m(x_i) = \left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$;
  $\mathrm{a_m} = \arg\min_{\mathrm{a},\rho} \sum_{i=1}^{N} \left[-g_m(x_i) - \rho h(x_i; \mathrm{a})\right]^2$;
  $\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} l(y_i, F_{m-1}(x_i) + \beta h(x_i; \mathrm{a}_m))$;
  $F_m = F_{m-1} + \nu\beta_m h(x; \mathrm{a}_m)$;
end

In our classification task, we convert $F(x_i)$ into a randomized predictor by using the soft-max function :

$$p(y_i = 1|x_i) = \frac{e^{F(x_i)}}{e^{F(x_i)} + e^{-F(x_i)}}$$

and use the Bernoulli log-likelihood for the loss function:

$$\begin{aligned} l(y, F) &= \log(\prod_{i=1}^{N} p(y_i = 1|x_i)^{y_i} p(y_i = 0|x_i)^{1-y_i}) \\ &= \sum_{i=1}^{N} \left[2y_i F(x_i) - \log(1 + e^{2F(x_i)})\right] \\ &= \sum_{i=1}^{N} l(y_i, F(x_i)) \end{aligned}$$

which results in:

$$l(y_i, F(x_i)) = 2y_i F(x_i) - \log(1 + e^{2F(x_i)}) \tag{7.10}$$

and g_m in Eq. 7.6 is obtained with:

$$g_m(x_{_i}) = 2(y_i - p_m(y_i = 1|x_i)) \tag{7.11}$$

and Eq. 7.8 is converted to:

$$\begin{aligned} \beta_m &= \arg\min_{\beta} \left\{ \sum_{i=1}^{N} 2y_i \left(F_{m-1}(x_i) + \beta h(x_i; \mathrm{a}_m)\right) \right. \\ &- \left. \sum_{i=1}^{N} \log\left(1 + e^{2(F_{m-1}(x_i) + \beta h(x_i; \mathrm{a}_m))}\right) \right\} \end{aligned}$$

The pseudocode for the gradient boosting algorithm is given in the following.

Algorithm Gradient boosting with Bernoulli log-likelihood loss function

$p_0(y_i = 1|x_i) = 0.5, F_0(x_i) = 0, \ \forall i$;
for $m \leftarrow 1$**to** M **do**
  $g_m(x_i) = 2(y_i - p_{m-1}(y_i = 1|x_i)), \ \forall i$;
  $h = \arg\min_{\hat{h},\mathrm{a}} \sum_{i=1}^{N} (-g_m(x_i) - \hat{h}(x_i; \mathrm{a}_m))^2$;

$$\begin{aligned} \beta_m &= \arg\min \left\{ \sum_{i=1}^{N} 2y_i \left(F_{m-1}(x_i) + \beta h(x_i; \mathrm{a}_m)\right) \right. \\ &- \left. \sum_{i=1}^{N} \log\left(1 + e^{2(F_{m-1}(x_i) + \beta h(x_i; \mathrm{a}_m))}\right) \right\} \end{aligned}$$

  $F_m = F_{m-1} + \nu\beta_m h(x; \mathrm{a}_m)$;
  $p_m(y_i = 1|x_i) = \frac{e^{F_m(x_i)}}{e^{F_m(x_i)} + e^{-F_m(x_i)}}, \ \forall i$;
end

After initialization, we calculate h and β to update the new F. This procedure is continued until a certain number of iterations M is reached. To prevent the overfitting or underfitting problems, we select optimum values of M and ν in a cross-validation test.

7.3 Selection of the Base Learner

The function h can have any form that can be optimized over the parameter a to fit the training data. In this paper we consider two kinds of base learner as follows:

7.3.1 Ordinary Least Squares (OLS) Base Learner

The simplest function to use here for h is the OLS regressor as:

$$h(x) = \alpha_1 x + \alpha_2 = \mathrm{aX}$$

where

$$\mathrm{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}, \ \mathrm{X} = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

By solving the Ordinary Least Squares (OLS) regression, we find the parameter $\mathrm{a}= [\alpha_1; \alpha_2]$ in Eq. 7.7:

$$\mathrm{a} = (\mathrm{X}\mathrm{X}^{\mathrm{T}})^{-1}\mathrm{X}^{\mathrm{T}}\mathrm{g}$$

7.3.2 Kernel-based Base Learner

Kernel based regression methods are considered as the problem of finding the function f that minimizes the objective function

$$\min_{f\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i) + \mu||f||^2 \tag{7.12}$$

As we explained in chapter 2, $\mathcal{H}$ is the Reproducing Kernel Hilbert Space (RKHS) generated by the kernel $k(.,.)$ and μ is a parameter that trades off the quality of the regression function and the regularization term. According to the representer theorem (Kimeldorf et al. [80]), explained in chapter 1, the optimal $f(x)$ has the form:

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x) \tag{7.13}$$

Recently different research has been done to use the idea of kernel in the boosting procedure [123, 23, 40] based on the representer theorem.

Similar to Eq. 7.5 in which $F(x)$ is a summation of base learners, $f(x)$ (Eq. 7.13) is also a summation of kernel functions. Compared with the other works [123, 23], we use a simple base learner to bring the concept of the kernel function in the boosting procedure. We consider:

$$h(x, \gamma) = \alpha_1 k(x, x_\gamma) + \alpha_2 = \mathrm{aK}$$

where

$$\mathrm{a} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix},\ \mathrm{K} = \begin{bmatrix} k \\ 1 \end{bmatrix}$$

and again we use OLS regression to find the parameters a and γ.

$$(h, \gamma) = \arg\min_{\hat{h},\mathrm{a},\gamma} \sum_{i=1}^{N} \Big(-g(x_i) - \hat{h}(x, \gamma, \mathrm{a})\Big)$$

Optimizing the parameter γ means that each kernel function is selected once at most, and this increases the run time of the algorithm. This also guarantees that the effect of some kernel functions is not excessively magnified and so prevents over-fitting in the boosting procedure.

Method	TREE	Spec. (%)	Sen. (%)	Acc. (%)	MCC.
OLS regression	Ficus	93.0	87.5	86.3	0.77
OLS regression	Bamboo	97.1	95.0	**96.1**	0.91
OLS regression	Schefflera	84.0	94.0	91.0	0.80
Kernel regression ($\sigma = 0.1$)	Ficus	88.1	93.0	91.1	**0.81**
Kernel regression ($\sigma = 0.1$)	Bamboo	99.1	80.0	86.6	0.80
Kernel regression ($\sigma = 0.1$)	Schefflera	75.0	95.0	88.6	0.75
Kernel regression ($\sigma = 10$)	Ficus	92.1	96.0	**95.1**	0.79
Kernel regression ($\sigma = 10$)	Bamboo	99.1	93.0	95.3	**0.93**
Kernel regression ($\sigma = 10$)	Schefflera	84.0	96.0	**92.9**	**0.82**

Table 7.1: The Performance of classification in sonar signals using the gradient boosting method

7.4 Experiment and Results

We gathered the sonar data, 720 echoes, each one 10000 data points, for each tree shown in Fig. 5.4. After preprocessing, those echoes were converted to 720 matrices of features, where in Eq. 7.1, $K = 1980$ (number of features), $C = 20$ (number of channels) and $S = 99$ (number of data points or frames in each channel). So, in our boosting procedure, N, the number of examples, is 720 and the number of features is 1980. But in training and in each boosting step, instead of using all features at once, we use only the features in each channel ($S = 99$) and find the regressor h for that and repeat the boost step for all channels. Then, in testing we use the corresponding regressor (base learner) of all channels to estimate the final $F(x)$. In all experiments the maximum number of iterations of the boosting algorithm, M, and the shrinkage factor, ν, were set to 100 and 1, respectively. For the kernel regression we chose the Radial Basis Function (RBF) kernel with different values of σ. After using 5-fold cross validation, the prediction quality was then evaluated by specificity (Spec.), sensitivity (Sen.), accuracy (Acc.) and also Matthew's correlation coefficient (MCC.). Table 7.4 shows the results of the classifier. As we see from that table, the gradient boosting with kernel-based base learner shows a slight improvement in accuracy and MCC. compared with the one that uses the OLS base learner, and also the value of σ in RBF kernels affects the performance of the classifier.

In the previous chapter, we used the optimal selection of time-resolved spectrum kernels (table 6.1) and it was shown an improvement compared with the previous work of our group (Wang et al. [135, 134]), in which the classification was made through template matching in 2D sonar acoustic image using a 2D Discrete Cosine Transform. Comparing the Tables 7.4 and 6.1 one sees that the boosting method could improve the performance of classification.

The other point is the running time of the boosting method. Table 7.4 shows the running time of the boosting and spectrum kernel methods on a PC with an Intel Core Duo processor (1.83GHz and 1GB RAM) while coded in Matlab 7.0 and in a 5-fold

cross validation test. From this table, we see that the boosting approach is more efficient than the spectrum kernel for the classification of sonar data and needs much less time for training.

	Training time	Testing time
Spectrum kernel	6 hours	2 mins
OLS regression Boosting	6 mins	10 seconds
Kernel regression Boosting	10 mins	20 seconds

Table 7.2: Running time of the proposed boosting method and Time-resolved spectrum kernel in a 5-fold cross validation test.

7.5 Conclusion

In this chapter, we proposed a kernel based boosting method for biosonar object classification. We used a regression approach in the gradient boosting which proved to be both more accurate and efficient than other previously proposed methods such as spectrum kernels and template matching using acoustic images. We suggested a simple base learner in the boosting method using the kernel matrix and showed that it outperformed the simple OLS regression. The main point of the signal preprocessing in our method, for biosonar classification, is using a filter bank like that of the hearing system of bats. With this filter bank, the one-dimensional sonar echoes are converted into shorter length but more informative multi-dimensional signals. After this conversion, the features are more distinguishable and the boosting method was able to classify them efficiently and to get satisfying results. Without this filter bank, the boosting method can not classify the raw echoes accurately.

Chapter 8

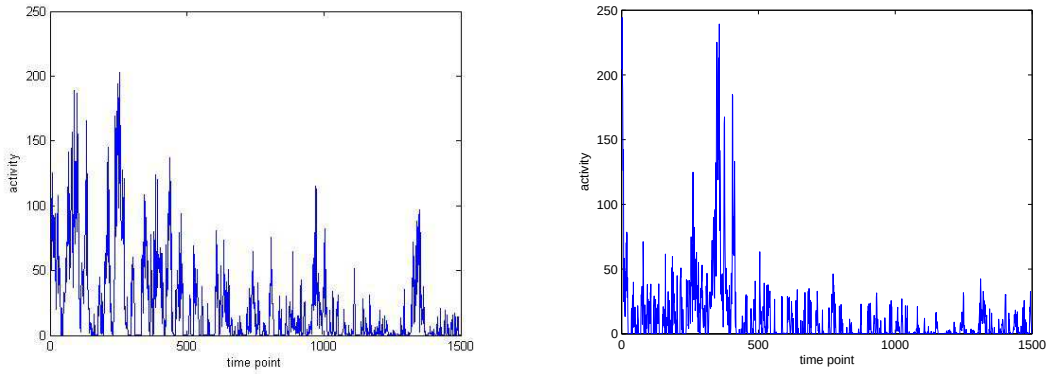
FIR–based Classifiers for Animal Behavior Classification

8.1 Introduction

The Forced Swimming Test (FST) is a behavioral test used frequently to evaluate the potential efficacy of drugs affecting the central nervous system (CNS) in rats or mice [104]. In this experiment, rats are exposed to a 15-min pretest swim period and followed the next day by a 5-min test swim. Immersion of rodents in water for an extended period of time produces a characteristic behavior called immobility, in which the rat makes only those movements necessary to keep its head above water. When antidepressant drugs are administered between the pretest and test periods, usually three times within 24hr, the behavioral immobility is selectively decreased [24]. Depending on the type of drug, rats show a mixed behavior of activities such as immobility, struggling/climbing (the rat tries to escape from the water) and swimming. Researchers have tried to conclude the effect of drugs from the above three states (immobile, struggling and swimming). Typically, tricyclic antidepressants and drugs with selective effects on noradrenergic transmission increase struggling/climbing behavior, while selective serotonin reuptake inhibitors increase swimming behavior versus the control group [42, 36, 37].

Fig. 8.1 shows examples of activity profiles, which are gained from successive images of rat movement in FST test. Considering predefined thresholds for the immobility, struggling and swimming states, and depending on the amplitude of the activity profile, the average period of time in each state can be measured. By comparing those parameters for the animals treated with an antidepressant against the control group, which was treated with the vehicle, it can be observed, for example, whether the swimming behavior of rats with an antidepressant drug has increased.

In an automated classification method, we aim to classify animals treated with known antidepressants and the control group. However, our experiments show that the response of the rat to drugs is too complex to only consider those states as indicator of the drug efficacy. For example, consider predefined thresholds in an activity profile with an arbitrary



(a) Tylose (b) Imipramine 40 mg

Figure 8.1: FST test as a behavioral test. The successive images of rat movement are converted to activity profile signals. a) activity profile (arbitrary unit) for tylose as control and b) Imipramine 40mg as antidepressant.

unit for the amplitude as follows:

- If amplitude of the activity profile < 15 then immobility state.
- If amplitude of the activity profile > 40 then struggling state.
- If amplitude of the activity profile $\in [15, 40]$ then swimming state.

It means that we have quantized the activity profile only in three states, in which the other values of the activity profile are not considered. This means, we have lost the information hidden in the signal and so the accuracy of classification. Even if we increase the number of states (quantization levels), the value of threshold at those states may affect the concluded results.

The detection of the behavior of rats depends on recognizing changes in some characteristics of movement and we are interested in features which represent also the dynamic behavior of rats.

System identification is a vital problem in many fields of biological modelling. It is mainly concerned with the determination of an input-output mapping of the system. It is the experimental approach to the modelling of a process or a plant of unknown parameters. Once the mathematical model is chosen, it can be characterized in terms of suitable descriptions such as transfer function, impulse response or power series expansions.

Here, we consider the activity profiles as outputs of a system and select a suitable model which extracts the dynamic behavior of those activity profiles [17]. One possible representation of the local dynamic behavior in the activity profile is using an Autoregressive (AR) model, in which future values are predicted from a combination of previous sample values. A few tens of seconds of a profile activity can normally be described by using up to ten AR coefficients and then the value of these AR coefficients as features can

be used in a classifier to classify those activity profiles. In this paper we show that these features are suitable for the automated classification.

Another issue in the automated classification is the presence of noise in activity profiles. We consider the activity profile, $x(n)$, as the total of the inherent response of a rat to a drug at a certain dose, $s(n)$, and the undesired and interference noise, $N(n)$, so we have:

$$x(n) = s(n) + N(n) \tag{8.1}$$

$N(n)$ is the undesired part, which affects the accuracy of classifier. The presence of this noise can be due to the experimental setup, the difference between the physiological behavior of rats, and so on. One method to remove this noise is to use a suitable filter based on its frequency content. But the frequency content of this noise is unknown. To resolve this problem, we use the fact that when a suitable filter is added to a classifier, it should increase the signal to noise ratio and so the accuracy of the classifier. For this, we use a general model of filters known as FIR (Finite Impulse Response):

$$\mathrm{W}_n = [w_n(0),\ w_n(1),\ ..., w_n(p-1)]^T \tag{8.2}$$

and the output of that filter is the estimation of $s(n)$:

$$\hat{s}(n) = \mathrm{W}_n^T x(n) \tag{8.3}$$

where the coefficients of the FIR filter are obtained via optimizing a criterion showing the accuracy of a classifier which tries to classify two different classes of signals.

As we explained in chapter 6, Kernel Fisher Discriminant Analysis (KFDA) has been used for classification and also for optimal kernel selection [92, 13, 14, 108]. In this study we use the Fisher discriminant criterion in the kernel space as a criterion for the accuracy of the classifier and try to find the optimal coefficient of the FIR filter that maximizes that criterion. To solve the optimization problem, we use the DIviding RECTangles (DIRECT) search method as an algorithm for global nonlinear optimization. We use Support Vector Machines (SVMs) for the classification task. Our proposed behavior classification method provides a reliable discrimination of different classes of antidepressant drugs (tricyclic imipramine and desipramine) administered to rats versus a vehicle-treated group [17].

In the following sections we discuss the autoregressive model and the suggested FIR based classifier in detail and then we explain the experimental setup and results of classification. At last, we show our conclusions.

8.2 Autoregressive models

Autoregressive modelling utilizes the time history of a signal to extract important information hidden in the signal. AR modelling is an alternative to the Discrete Fourier Transform (DFT) in the calculation of a power spectrum density function and frequency content of

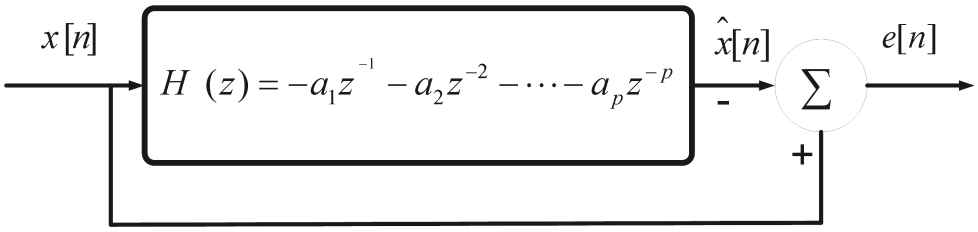


Figure 8.2: AR analysis filter. $x[n]$ and $\varepsilon[n]$ represent input and output sequences, respectively and $a_1 ... a_p$ are predictor coefficients.

a time series. In biomedical applications, AR modelling is notably used in the dynamic behavior analysis of heart rate [98] and electroencephalogram recordings [44]. A review of classification algorithms for EEG-based brain-computer interfaces [44] also shows that the AR modelling has been widely used by researchers of this field, *e.g.*, mental tasks classification [29] and brain–computer interface design [62].

Basically, the AR modelling of a time series is based on an assumption that the most recent data points contain more information than the other data points, and each value of the time series can be predicted as a weighted sum of the previous values plus an error term. The model is defined by :

$$x[n] = \sum_{i=1}^{p} a_i x[n-i] + \varepsilon[n] \tag{8.4}$$

where $x[n]$ is the current value of the time series, $a_1 ... a_p$ are predictor (weighting) coefficients, p is the model order (the number of previous points to be considered for the prediction of the new data point), and $\varepsilon[n]$ represents a one-step prediction error. It determines an analysis filter, through which the time series is filtered (Fig. 8.2). The predictor coefficients are usually estimated using the least-squares minimization technique such that the sum of squares of $\varepsilon[n]$ is minimized. Two solutions to AR modelling are the Yule-walker equation [138, 133] (autocorrelation method) and the recursive implementation of that [87].

Fig. 8.3 shows an example of activity profiles (Imipramine 40mg) and its estimation, $\hat{x}(n) = \sum_{i=1}^{p} a_i x[n-i]$, through the AR model. AR coefficients are used as features in our classification task.

8.2.1 Selecting the Model order

If the order of the AR model is selected to be small, the spectrum will be smoothed, resulting in resolution loss. In contrast, if the order is selected to be large, the spectrum will contain spurious peaks. The solution is to increase the order p until the modelling error is minimized. Thus, in order to overcome this problem, a penalty function $C(p)$, which increases linearly with the model, is introduced:

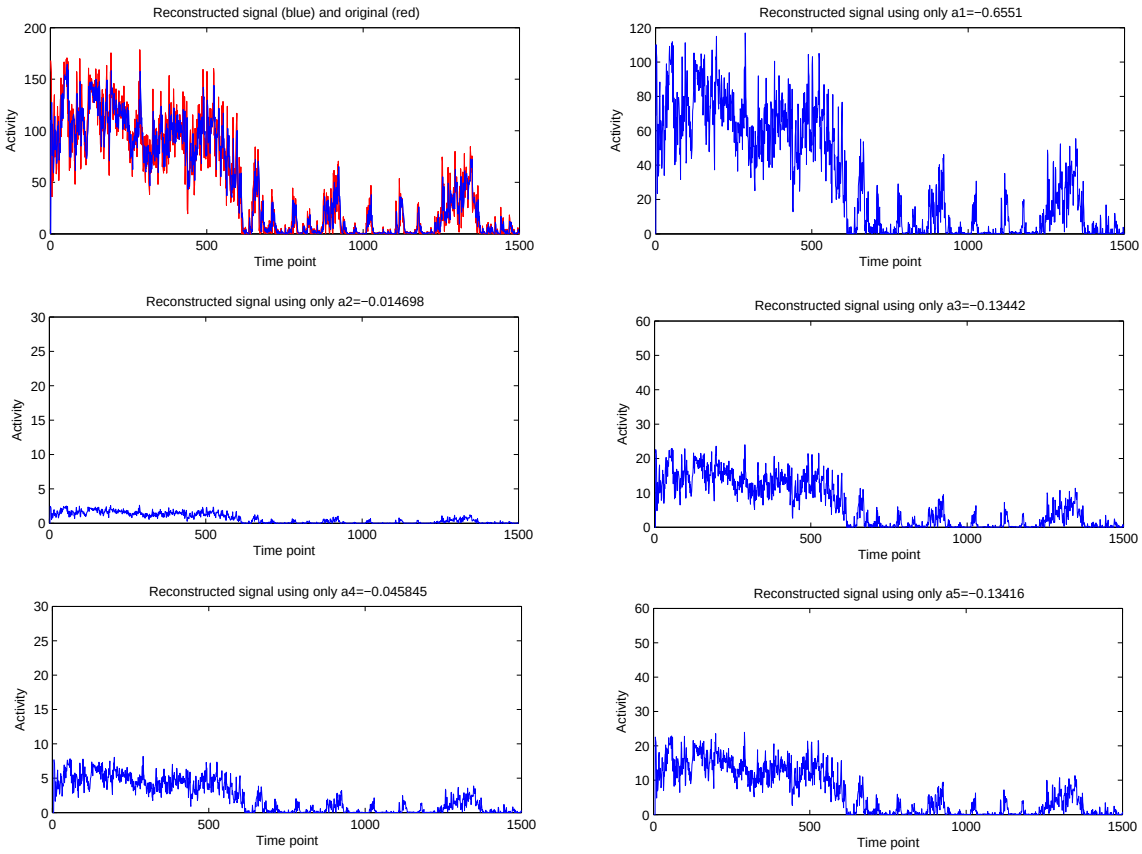


Figure 8.3: AR coefficients of an activity profile of imipramine 40 mg. $p = 5$ is the model order and $a = [-0.6551, -0.014698, -0.13442, -0.045845, -0.13416]$ are AR coefficients. a) The original signal. b)-f) Reconstructed signals and effect of each coefficient on estimated $\hat{x}(n)$.

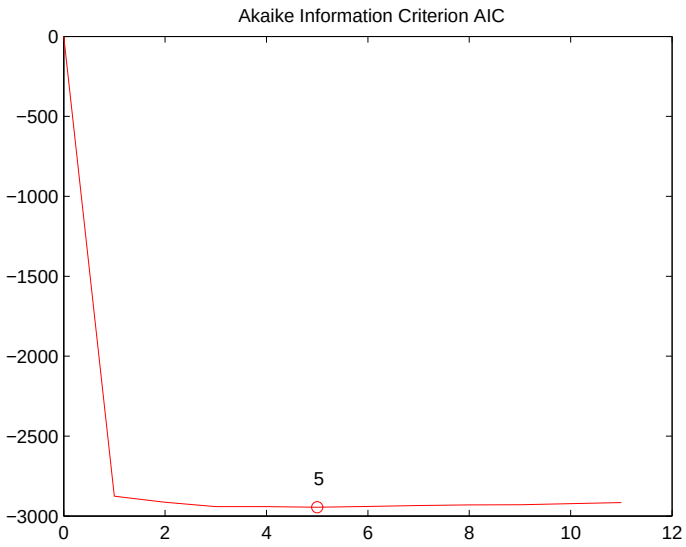


Figure 8.4: Akaike's Information Criterion (AIC) values for model orders of 1 to 12 for an instance of activity profile (Fig. 8.3). The optimum value of the order is $p_{opt} = 5$.

$$C(p) = N \log \varepsilon_p + f(N)p \tag{8.5}$$

where ε_p is the modelling error; N is the length of the data record, and $f(N)$ is a function of N. The key idea is then to select the value of p that minimizes $C(p)$. one proposed selection criteria is Akaike's Information Criterion (AIC) [1]:

$$AIC(p) = N \log \varepsilon_p + 2p \tag{8.6}$$

This criterion represents a trade-off between the estimated prediction error and the size of the model. Fig. 8.6 plots the AIC value vs. model order for the activity profile shown in Fig. 8.3. For this example, we see that the optimum value of the order is $p_{opt} = 5$.

8.3 Algorithms

8.3.1 FIR filter based classifier

We consider the problem of classification of signals that contain additive unknown or undesired parts interference (noise). If we can not find suitable features representing the desired part of signals or if we can not deduce the interference signal from the original signal, the classifier may face overfitting and so a decrease of the accuracy.

FIR filters are designed to filter out the undesired part from the signal based on its frequency specification. Because of its stability and simplicity in implementation, they

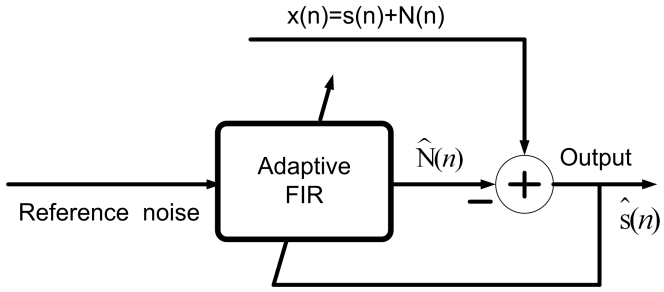


Figure 8.5: Adaptive filter for noise cancellation

are used frequently in different applications. However, in some applications (such as forced swim test classification), we do not have the frequency specification of the noise, needed to design the FIR filter.

Fig. 8.5 shows the block diagram of an adaptive filter which is used for noise (interference) cancellation. As we know, if the reference noise is uncorrelated to the signal $s(n)$, the output of the FIR filter is an estimation of the interference ($\hat{N}(n)$) and the filter readjusts itself continuously to minimize the error between N and $\hat{N}$. The coefficients of the FIR filter can be obtained through the Least Mean Square (LMS) algorithm. But in our task the stationary and uncorrelatedness conditions for s and N are not met and we can not estimate $s(n)$ using the adaptive filter.

To cope with the problem, we suggest a new solution. We propose a FIR filter based classifier, in which the FIR filter tries to remove undesired parts by getting feedback from the accuracy of a following classifier. Fig. 8.6 shows our proposed solution. Consider $x_1(n) = s_1(n) + N_1(n)$ and $x_2(n) = s_2(n) + N_2(n)$ as two signals that contain undesired parts $N_1(n)$ and $N_2(n)$. Our aim is to estimate $s_1(n)$ and $s_2(n)$ as outputs of the added FIR filter. If we consider the l-length coefficient of the FIR filter as:

$$\mathrm{W}_n = [w_n(0),\ w_n(1),\ ..., w_n(l-1)]^T$$

we have:

$$\hat{s}_1(n) = \mathrm{W}_n^T x_1(n), \tag{8.7}$$

$$\hat{s}_2(n) = \mathrm{W}_n^T x_2(n) \tag{8.8}$$

The classifier should discriminate $\hat{s}_1(n)$ from $\hat{s}_2(n)$. The coefficients of the FIR filter that increase the accuracy of the classifier also increase the signal to noise ratio. Then, the task is to find optimum values of the coefficients that maximize a criterion showing the accuracy of the classifier. We select the Kernel Fisher discriminant criterion as a suitable objective for our optimization problem.

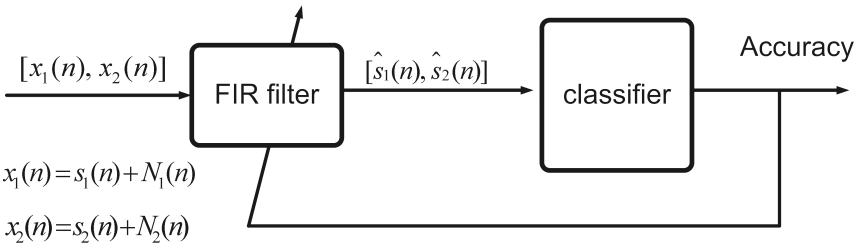


Figure 8.6: Optimal FIR filter based classifier. The coefficients of FIR filters are gained via optimization of a criterion showing the accuracy of the classifier

8.3.2 Kernel Fisher discriminant based optimal FIR filter

We want to find the coefficients of the FIR filter that maximizes the performance of the classifier. Here, we select Fisher discriminant criterion in the kernel space as a criterion for the performance of the classifier. As we discussed before in chapter 6, the kernel Fisher discriminant analysis is a non-linear extension of the linear Fisher discriminant analysis. It finds the direction in a feature space, defined implicitly by a kernel, onto which the projections of positive and negative classes are well separated in terms of the Fisher discriminant ratio. The more two classes are separated, the higher the performance of the classifier is. Then, we make a relation between that criteria and the FIR filter coefficients and find the coefficients that maximize that criterion. The kernel Fisher discriminant based optimal FIR filter tries to find the coefficients of the FIR filter that results in a vector v in feature space, on which projections of points give the maximum separation of the mean scaled in the feature space and the minimum variance in that direction (Fig. 8.7).

In chapter 6, we saw that the Fisher criterion in the kernel space can be represented in terms of a kernel matrix K, as:

$$J_{\max}(K) = y^T K(KD^T K + \lambda I)^{-1} K y \tag{8.9}$$

where:

$$y = \begin{bmatrix} (1/n_+)1_{n_+} \\ (-1/n_-)1_{n_-} \end{bmatrix}_{n\times 1} \tag{8.10}$$

in which λ is a regularization factor and 1_n and I_n denote the vector of all ones and the identity operator in $\mathbb{R}^d$, respectively.

The next step is to represent the kernel matrix in terms of the FIR coefficients. In Fig. 8.7, if we consider the l-length coefficient of the FIR filter as:

$$\mathrm{W}_n = [w_n(0),\ w_n(1),\ ..., w_n(l-1)]^T$$

we have:

$$\hat{s}_1(n) = \mathrm{W}_n^T x_1(n),\ \hat{s}_2(n) = \mathrm{W}_n^T x_2(n)$$

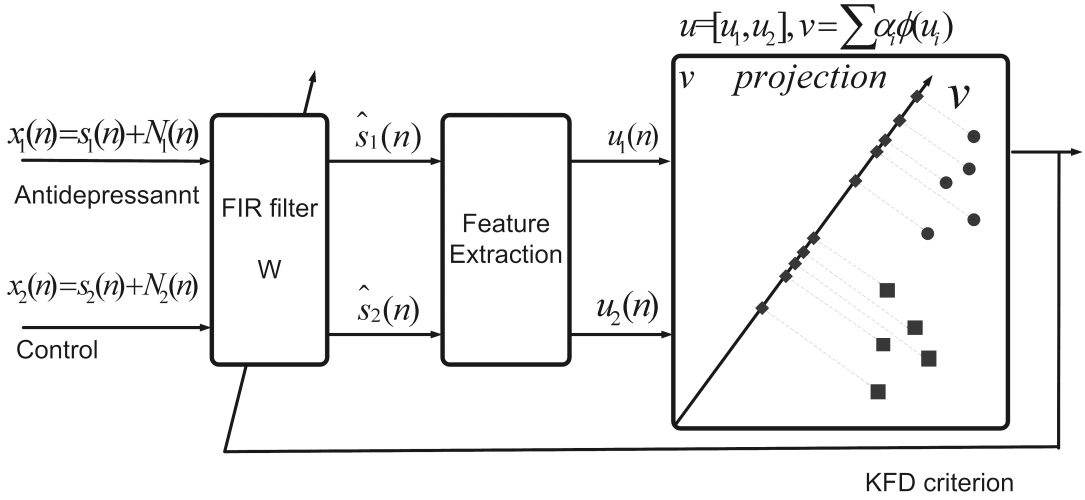


Figure 8.7: Kernel Fisher discriminant based optimal FIR filter tries to find the coefficients of the FIR filter that result in a vector v in feature space, on which projections of points give the maximum separation of the mean scaled in the feature space and the minimum variance in that direction.

We extract the features (AR parameters) u_1 and u_2 from $\hat{s}_1(n)$ and $\hat{s}_1(n)$ resp. and then:

$$K(u_1, u_2) =< \phi(u_1) * \phi(u_2) >$$

From the above equation and equation (8.9) we can say that when the coefficients of the FIR filter vary, the KFD criterion varies, too. However, the task is to find the optimum coefficients of the FIR filter that maximize the KFD criterion, *i.e.*, maximize the signal to interference ratio. For this, we use a method of global optimization, by which we can search for the optimum global value of the FIR coefficients in a given range.

8.3.3 Direct search

The DIRECT algorithm as an effective pattern search method was proposed by Jones et al. [73] for bound constrained global optimization. It deals with problems on the form

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & x_L \leq x \leq x_{U,} \end{aligned}$$

where $f \in \mathbb{R}$ and x, x_L, $x_U \in \mathbb{R}^n$.

The DIRECT algorithm is one of a class of deterministic direct search algorithms that does not require gradients. The objective function f must only be continuous in the neighborhood of a global optimum. It works by iteratively dividing the search domain into boxes that have exactly one function value at the box's center. At first, it transforms

the search space to be the unit hypercube. The function is then sampled at the center point of this cube. The hypercube is then divided into smaller hyperrectangles whose center-points are also sampled. In each iteration, through evaluation the objective function at those centers, the algorithm determines which boxes are most likely to have a better point than the current optimal one. A box is considered potentially optimal, if it has the potentially best function value for a given Lipschitz constant. The process continues after a prespecified number of function evaluations. The definition of *potentially optimal* from [73] follows:

Definition 8.1: *Suppose that the unit hypercube has been partitioned into m hyperboxes. Let c_i denote the center point of the ith hyperrectangle, and d_i denote the distance from the center to the vertices. Let $\epsilon > 0$ be a positive constant. A hyperrectangle j is said to be potentially optimal if there exists some $\hat{K} > 0$ such that:*

$$\begin{aligned} f(c_j) - \tilde{K} d_j &\leq f(c_i) - \tilde{K} d_i, \quad \text{for all} \quad i = 1, ..., m, \\ f(c_j) - \tilde{K} d_j &\leq f_{\min} - \varepsilon |f_{\min}| \end{aligned}$$

The formal steps of the DIRECT algorithm from [59] are given in the following. A detailed example of the search domain in the DIRECT algorithm was given in [136]. The serial and the parallel implementations of the algorithm have been discussed in [59] and [58], respectively.

8.4 Methods

8.4.1 Experimental setup

The experiments have been performed at the site of an industrial cooperation partner.

Animal and Drug specifications

Male Wistar rats (Charles-River Germany), weighing 180-220g, were used. They were housed in groups of 4 in an animal room with standard conditions ($20 - 22$°C, 50-55% relative humidity, 12h light/dark cycle with light on at 6.00a.m.). The rats, with freely available food and water, were left in the animal room for a minimum of 5 days to adapt to the new environment. Vehicle (0.5 %Tylose) or clinically used antidepressants of different classes suspended in vehicle were administered orally using a volume of 4ml/kg: imipramine (40mg/kg) and desipramine (30mg/kg; both tricyclic and antidepressants) [93, 125, 19]. For this work, 218 rats treated with vehicle, 72 treated with imipramine and 112 treated with desipramine were used. Drugs or vehicle were applied immediately after the pre-test session, 5hrs before the test session, and 1hr before the test session.

Algorithm DIRECT search method

Step 1 Normalize the search space to be the unit hypercube. Let c_1 be the center-point of this hypercube and evaluate $f(c_1)$, $m = 1$, and $t = 0$ (iteration counter).

Step 2 Identify the set S of potentially optimal rectangles [73].

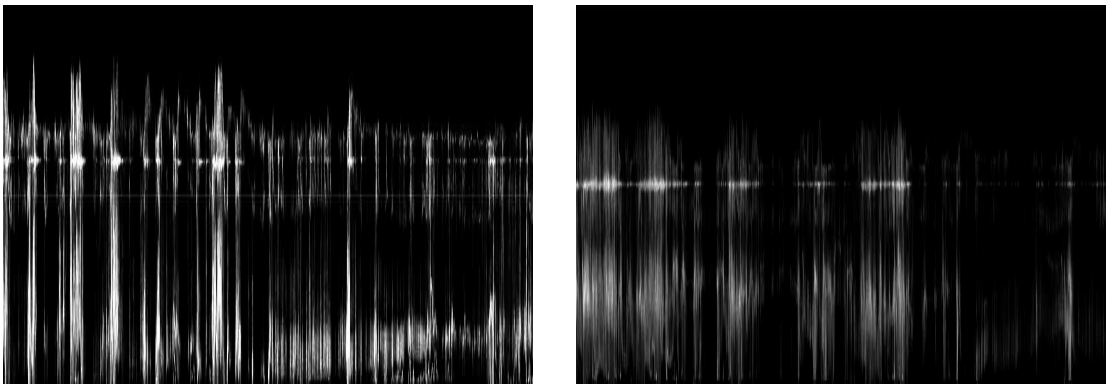
Step 3 Select any rectangle $j \in S$.

Step 4 Divide the box j as follows:

- Identify the set I of dimensions with the maximum side length. Let δ equal one-third of this maximum side length.
- Sample the function at the points $c \pm \delta e_i$ for all $i \in I$, where c is the center of the box and e_i is the ith unit vector.
- Divide the box j containing c into thirds along the dimensions in I, starting with the dimension with the lowest value of $w_i = \min\{f(c + \delta e_i), f(c - \delta e_i)\}$ and continuing to the dimension with the highest w_i. Update f_{min} and m

Step 5 Set $S = S - \{j\}$. If $S \neq 0$ go to Step 3.

Step 6 Set $t = t + 1$. If iteration limit or evaluation limit has been reached, stop. Otherwise, go to Step 2.



(a) Vehicle-treated (b) Desipramine-treated

Figure 8.8: Activity images of two representative rats treated with a) vehicle (0.5% tylose, p.o.) and b) desipramine (30mg/kg, p.o.)

Forced swimming test procedure

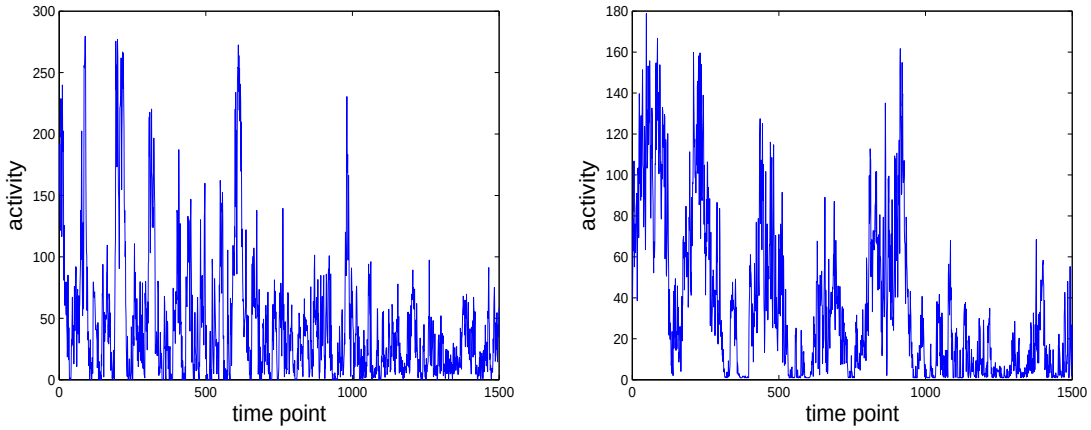
The forced swimming test procedure is as described in [104]. Briefly, each animal is placed to swim for 15min in a cylinder (height: 40cm; diameter:18cm) containing 18cm of water at 25°C (pretest session). They are then taken out and allowed to dry for 20min in a cage placed below an infrared lamp. Twenty-four hours after the pre-test session, they are again placed in the cylinder for 5min (test session), and the behavior of rats is recorded with a camcorder while it is assured that the camera lens and water line are on a horizontal line in order to minimize the area of distortion due to reflections on water surface.

Calculation of activity profiles

The image analysis software Halcon 7.0 (MVTec Software GmbH, Munich, Germany) was used to analyze the video tapes of rat movements. To extract the activity profile showing the movement of a rat, for five consecutive frames, the difference between the previous and the next image was calculated and binarized with a fixed threshold and then totalized into one gray level image. Within each activity image all non-zero pixels were summed up in the vertical direction to obtain one activity profile of the whole animal. Fig. 8.9 shows the corresponding activity profiles of Fig. 8.8.

8.4.2 Computational setup

We considered the FIR filter coefficients (W in Fig. 8.7) in the range of [-1,1] and the length of 10 for that. In order to reduce the dimension for our classification problem, we computed the AR parameters of each activity profile via the recursive solution of the Yule-Walker equation (Levinson method [87]). The optimum value of the order of the AR model, p_{opt} (EQ. 8.5), was gained though Akaike's Information Criterion (EQ. 8.6). In our



(a) Vehicle-treated (b) Desipramine-treated

Figure 8.9: Calculated activity profiles for two representative rats treated with a) vehicle (0.5% tylose, p.o.) and b) desipramine (30mg/kg, p.o.).

experiment, the average value of 5 was gained for p_{opt}. In the next step, we considered an RBF (Radial Basis Function) kernel for the Kernel fisher discriminant analysis with γ=1 ([31]). Then, we used the optimization method told above, to find the optimum values of the FIR filter coefficients, W_{opt}. In the next step, we used a SVM classifier with a linear kernel. We found the optimum C parameter of the SVM classifier using a simple grid search in the range of $[2^{-2}, 2^{15}]$ in terms of the maximum accuracy of the classifier.

We used a 5-fold cross validation for our classification task, *i.e.*, the whole dataset was split into 5 sets, of which 4 sets were used for finding the FIR filter coefficients and training the SVM classifier, and one set left out for testing. The procedure was repeated such that each set was used once for testing. The obtained results from the 5 folds then were averaged to produce a single estimation.

8.5 Results and discussion

We used the computational methods described above to classify rats treated with antidepressants of two different classes, tricyclic antidepressant (imipramine 40mg/kg: 72 rats; desipramine 30mg/kg: 112 rats), against a control group treated with vehicle (0.5% tylose: 218 rats).

The prediction quality was then evaluated by specificity (Spec.), sensitivity (Sen.), accuracy (Acc.) and also Matthew's correlation coefficient (MCC.).

To show the effect of the FIR filter on the classifier, we first do not use the FIR filter but only a SVM classifier with the RBF kernel. The optimum values of parameters γ of the RBF kernel and C in SVM classifier are gained through a grid search method and in terms of maximum average accuracy for the classifier with 5-fold cross validation. Tables

8.5 shows the results of the classifier.

Experiment	Spec. (%)	Sen. (%)	Acc. (%)	Mcc.
Imipramine 40mg	79.1	85.3	80.0	0.58
Desipramine 30mg	75.4	85.1	76.1	0.46

Table 8.1: Performance of SVM classifier with optimum parameters of γ and C (without optimum FIR filter) in classification antidepressant drugs vs. control.

Tables 8.5 shows the results of our new method with the FIR filter as described before. As we see, there is a notable improvement in performance with our method.

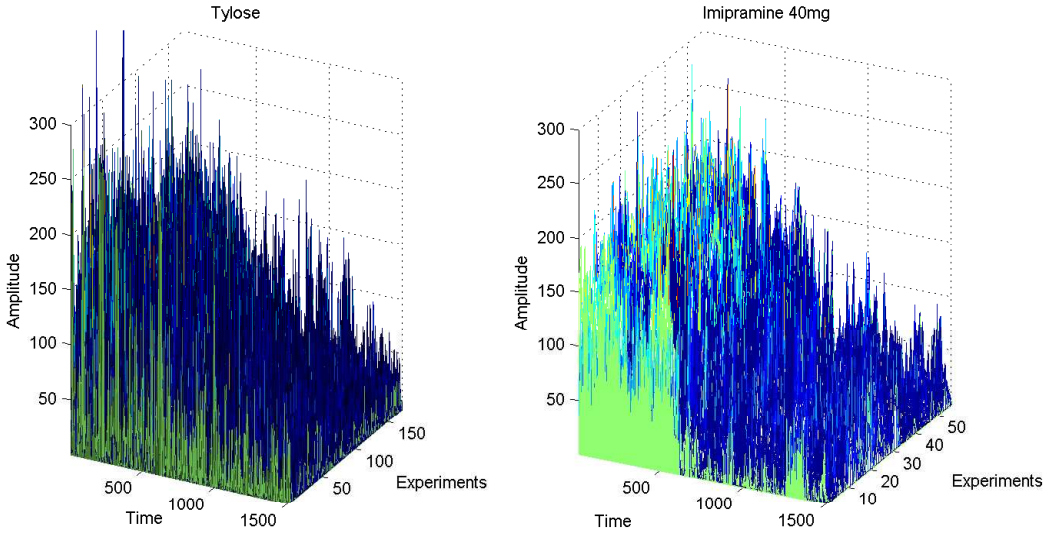
Experiment	Spec. (%)	Sen. (%)	Acc. (%)	Mcc.
Imipramine 40mg	93.1	93.3	93.3	0.84
Desipramine 30mg	98.0	81.7	86.6	0.75

Table 8.2: Performance of our method (optimum FIR filter) in classification antidepressants drugs vs. control.

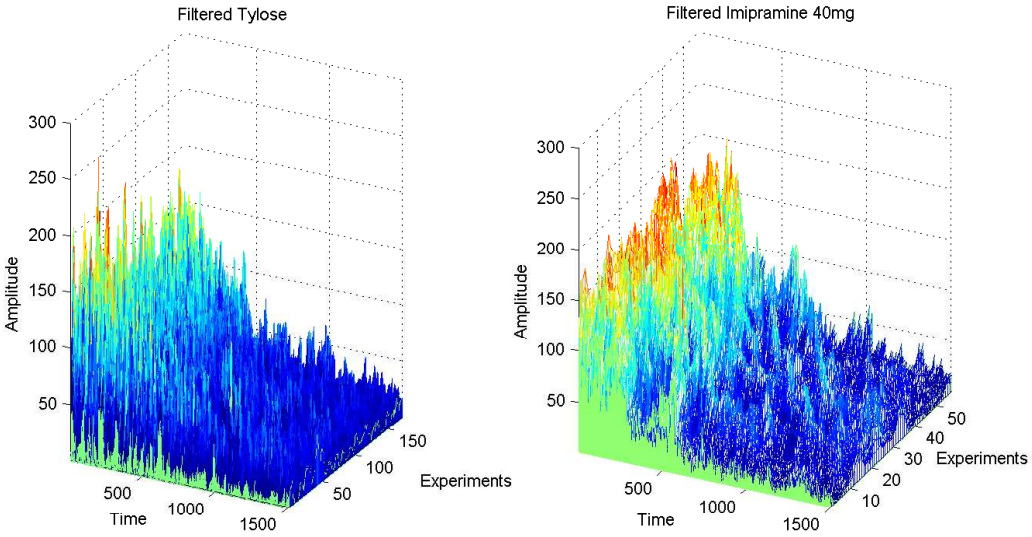
Fig. 8.10 shows the activity profiles of tylose and imipramine 40mg before and after filtering. It is apparent that the data of the filtered activity profiles are more distinguishable than that of the unfiltered activity profiles. In our method, the classifier extracts the features from the filtered activity profiles, while in the second method the SVM classifier works on the unfiltered data and tries to find the optimum values of γ and C parameters in terms of classification accuracy, and this may lead to overfitting and can result in an increased error rate for new unseen activity profiles. Fig. 8.11 shows the effect of the FIR filter on the feature space. In Fig. 8.11.b, in which we have used the FIR filter, the features are more discriminative compared with the features from the unfiltered activity profile which are influenced from the interference noise (Fig. 8.11.a). Fig. 8.12 shows the frequency response and phase of the obtained optimum FIR filter in the classification of imipramine 40 vs. control and also the optimization of the KFD criterion.

8.6 Conclusions

In this section, we implemented a new method for classification of biological signals in general, and in the forced swimming test as an example. The hypothesis behind our method is that if we can deduce the interference signal from the original signal, the accuracy of the classifier increases, otherwise, if our features are influenced from that interference signal, the classifier faces overfitting and can not classify them accurately. We used a FIR filter to filter out those additive noise from the signal. The parameters of the FIR filter were obtained via maximizing the accuracy of a classifier that tries to make discrimination between two classes of the activity profiles (e.g. drug vs. control). We used the kernel Fisher discriminant as a criterion for the discrimination and the DIRECT

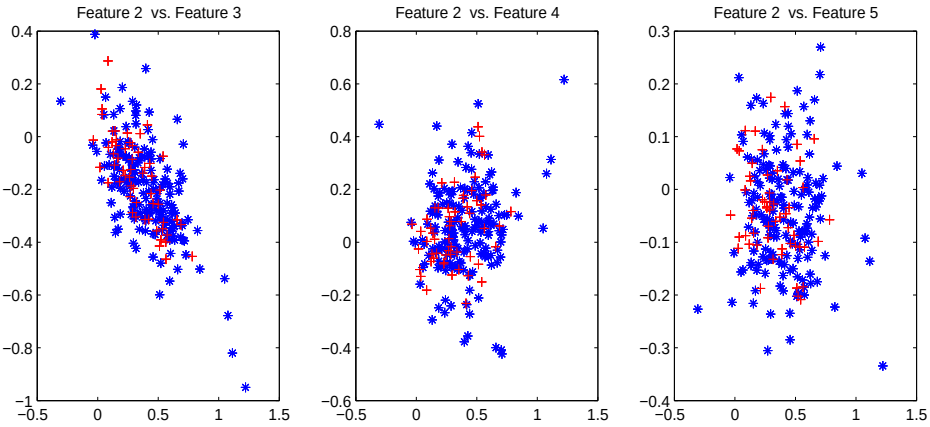


(a) Activity profiles

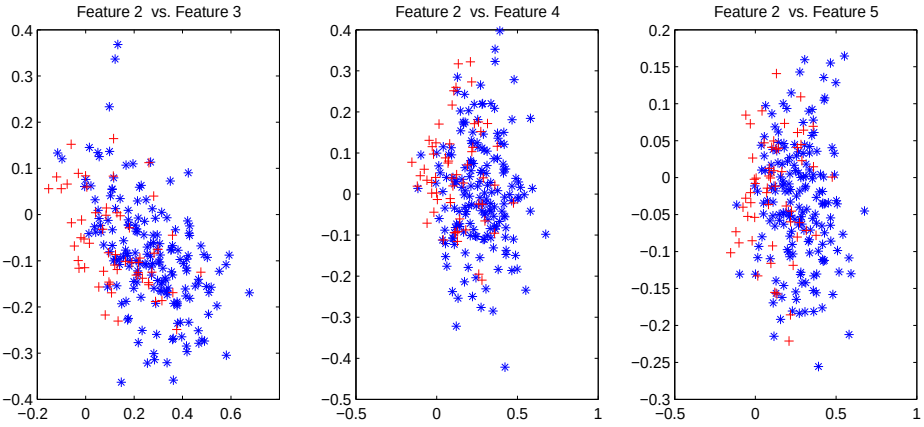


(b) Activity profiles after the obtained FIR filter

Figure 8.10: The effect of the optimal FIR filter on the activity profiles of tylose and imipramine 40mg. a) Raw data of activity profiles. b) Filtered activity profiles.

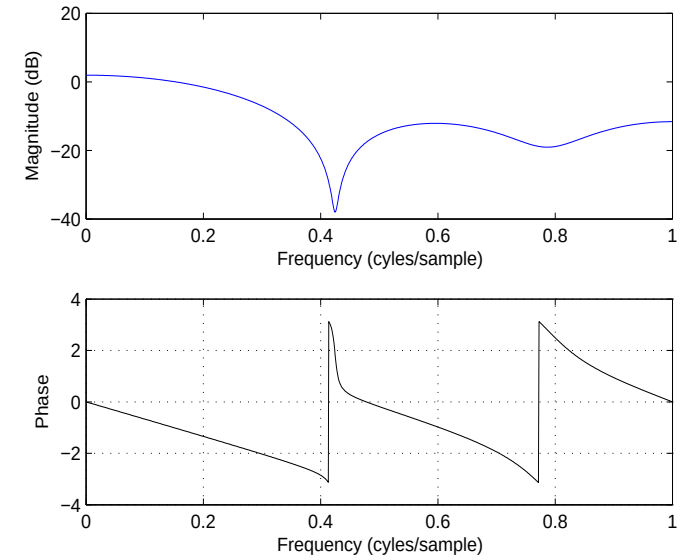


(a) AR features of an instance of activity profile

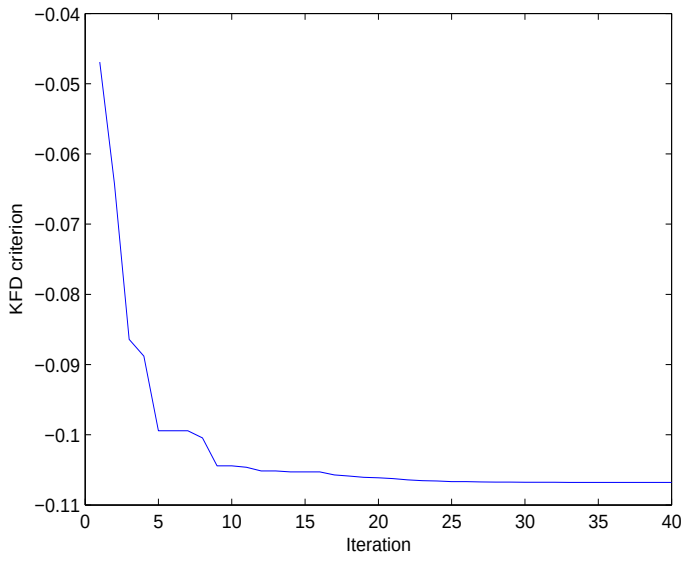


(b) AR features of the activity profile after FIR filter

Figure 8.11: The effect of the obtained optimal FIR filter on the AR features of an instance of activity profile of tylose (blue *) and that of imipramine 40mg. (red +). a) without FIR filter. b) with FIR filter.



(a)



(b)

Figure 8.12: Classification imipramine 40 vs. control. a) Frequency response and phase of the obtained optimum FIR filter. b) KFD criterion optimization

search method for solving the optimization problem. Our proposed behavior classification method allowed for a very reliable discrimination of rats subjected to different classes of antidepressant drugs (tricyclic imipramine and desipramine) versus a vehicle-treated group. We suggested AR parameters as suitable features for extraction of the dynamic behavior of rats in the forced swimming test. Using these features, we do not need to quantize the activity profile to 3 states of immobility, struggling/climbing and swimming, and hence we do not loose the valuable information needed for classification. We showed that these features can be used in the study of the effect of drugs in rats. Furthermore, We believe, with some modifications, our proposed FIR based classifiers can also be used in other biological applications such as EEG signals classification and Brain Interface Control (BCI).

Chapter 9

Summary

In this thesis, the main task was to find robust, fast and precise learning methods for noisy, incomplete, with very limited amounts of data, while considering the structure and dimension of data. Kernel methods, founded on strong theoretical grounds, operate on all types of data and provide a unified framework to interpolate between pattern analysis, signal processing and string processing. We selected them as main pattern recognition methods and our goal was to extend and improve them for the problems at hand.

A kernel method solution comprises two parts: a module that performs the mapping into the feature space and a learning algorithm, designed to discover linear patterns in that space. The first class of methods that implemented the theory were Support Vector Machines (SVMs). SVMs represent a very specific class of algorithms, characterized by the use of kernels, the absence of local minima, the sparseness of the solution and the capacity control obtained by acting on the margin or on other dimension independent quantities as the number of support vectors. In Section 3.1, we presented some examples from different applications of system biology problems and bioinformatics, showing that SVMs are currently the best performing methods in various domains. We think this work also showed some usefulness of kernel based pattern analysis methods in a broad range of high dimensional biological data. We briefly summarize the main results of this thesis:

In chapter 4, our aim was to develop an accurate method for classification of GPCR families, especially at the sub-subfamily level, at which we have the problem of imbalanced data. We chose the local alignment kernel [64] as a suitable kernel for our classification task. To solve the data imbalance problem, we suggested a new approach of oversampling for the imbalanced protein data in which the minority class in the data space is oversampled by creating synthetic protein sequences, considering the distribution of the minor and major classes. Using the local alignment kernel along with our oversampling technique, we could get better accuracy and Matthew's correlation coefficient for the classification of GPCRs at the subfamily and sub-subfamily level than other previously published method [22, 74]. We also developed a systematic study using GPCRs as a set of real and artificially generated datasets to show the efficiency of our method and how the degree of class overlapping can affect class imbalance. The results showed

that our SPSO algorithm outperformed other oversampling techniques. In summary, this method can be used for protein classification problems and remote homology detection, where classifiers must detect a remote relation between unknown sequence and training data with an imbalance problem.

In chapter 5, we studied the classification of biosonar signals as an example of the random process signals which contain local similarities. We suggested a kernel called *time-resolved spectrum* kernel to measure the similarity of echoes as time series and made a relation between that kernel and geometric specification of the objects. The p-length subsequence of that kernel simply measures the occurrences of fixed p-length subsequences for each of the time series in consideration. The more time series share similar p-length subsequences, the more similar they are. We also proposed a more general kernel called *warped time-resolved spectrum* kernel, which considers warping in the subsequences. We then used those kernels directly in a SVM-based classifier. We saw that by changing the warping cost parameter the accuracy of classifier changes. This parameter lets the kernel consider a warping (with a cost) for the subsequences of the time series and extract their similarity. Considering that parameter in our classification task was justifiable because the echoes reflected by the adjacent leaves of each tree can have somehow similar patterns but not exactly the same, so we need to have a parameter that can let the kernel capture those similarities, too. The optimal value of that parameter for each tree can be related to the physical specification of each tree. Our results provided evidence that this kind of kernel can be used for pattern extraction and classification in random signals. We think this kind of kernel is suitable for pattern recognition in signals with inherent self similarity and for estimating periodicity in arbitrary time series like speech and biomedical signals.

In chapter 6, having a set of the kernels suggested in chapter 5, we proposed a new method to find an optimal linear combination of those kernels. We formulated the optimal kernel selection via maximizing the Kernel Fisher Discriminant criterion (KFD) and used the Mesh Adaptive Direct Search (MADS) method to solve the optimization problem. This optimization method needed less run time than the other suggested optimization method [108] that brings the objective function in the form of Semi-Definite Programming(SDP) via the Schur complement technique. We obtained better results with our suggested kernel selection method compared with other matching methods [135, 134]. Despite the accurate rate of classification, the main drawback of the time-resolved spectrum kernel was the low speed of both training and testing procedures. It prevented us to use the method for real–time applications.

In chapter 7 we saw how the boosting method, when it is used with the kernel functions, can gives satisfying results with much less run time than the time-resolved spectrum kernel. In this chapter, we presented an algorithm based on gradient boosting for biosonar signal classification. We presented two kinds of base learners for the gradient boosting: Ordinary Least Squares (OLS) and kernel-based base learners. The main point of the signal preprocessing in our method, for biosonar classification, was using a filter bank like that of the hearing system of bats. With this filter bank, the one-dimensional sonar echoes were converted into shorter length but more informative multi-dimensional signals. After

this conversion, the features are more distinguishable and the boosting method was able to classify them efficiently and to get satisfying results. Compared with our previous works in chapters (5 and 6), we got more efficient and accurate results with the newly proposed boosting method, which made it feasible for the real applications.

In chapter 8 we implemented a new method for classification of biological signals in general, and used it in the animal behavior classification as an example. We used a Finite Impulse Response (FIR) filter to filter out the additive noise from the activity profile. The parameters of the FIR filter were obtained via maximizing the accuracy of a classifier that tries to make a discrimination between two classes of the activity profiles (e.g. drug vs. control). We used the kernel Fisher discriminant criterion as a criterion for the discrimination, the DIviding RECTangles (DIRECT) search method for solving the optimization problem and Support Vector Machines (SVMs) for the classification task. We showed that Autoregressive (AR) coefficients are suitable features for the extraction of the dynamic behavior of rats and also the classification of activity profiles. Our proposed behavior classification method provided a reliable discrimination of different classes of antidepressant drugs (imipramine and desipramine) administered to rats versus a vehicle-treated group. We believe, with some modifications, our suggested FIR based classifiers can also be used in other biological applications such as EEG signals classification and Brain Interface Control (BCI).

Bibliography

[1] H. Akaike. A new look at the statistical model identification. In *IEEE Trans. Automat. contr.*, volume 22, pages 203–217, 1974.

[2] A. Al-Shahib, R. Breitling, and D. Gilbert D. Feature selection and the class imbalance problem in predicting protein function from sequence. *J. Appl. Bioinformatics*, 4(3):195–203, 2005.

[3] S. F. Altschul, T. L. Madden, A. A. Schaffer, Z. Zhang, W. Miller W, and D. J. Lipman. Gapped blast and psi-blast: A new generation of protein database search programs. *J. Nucleic Acids Res.*, 25:3389–3402, 1997.

[4] T. K Attwood, M. D. R. Croning, and A. Gaulton. Deriving structural and functional insights from a ligand-based hierarchical classification of g-protein coupled receptors. *J. Protein Eng.*, 15:7–12, 2002.

[5] C. Audet and J.E Dennis Jr. Analysis of generalized pattern searches. *SIAM J. optim.*, 13:889–903, 2003.

[6] C. Audet and J.E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. optim.*, 17(1):188–217, 2006.

[7] E. Ayrapetyants and A. I. Konstantinov. *The Elements of Statistical Learning*. Joint Publications Research Service, Arlington, Virginia, 1979.

[8] A. Bairoch and R. Apweiler. The swiss-prot protein sequence data bank and its supplement trembl. *Nucleic Acids Res.*, 29:346–349, 2001.

[9] Bottou L. Bakir, G. H. and and J. Weston. Breaking SVM Complexity with Cross–Training. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 81–88, Cambridge, MA, USA, 2005. MIT Press.

[10] M. M. Beigi, M. Wang, and A. Zell. Time-Resolved Spectrum Kernel for Biosonar Target Classification. In *The Fourth IASTED International Conference on Signal Processing, Pattern Recognition, and Applications (SPPRA 2007)*, pages 126–131, 2006.

[11] M. M. Beigi and A. Zell. A Novel Method for Classifying Subfamilies and Subsubfamilies of G-Protein Coupled Receptors. In *International Symposium on Medical Data Analysis (ISBMDA),Thessaloniki, Greece*, pages 25–36, 2006.

[12] M. M. Beigi and A. Zell. SPSO: Synthetic Protein Sequence Oversampling for Imbalanced Protein Data and Remote Homology Detection. In *International Symposium on Medical Data Analysis (ISBMDA),Thessaloniki, Greece*, pages 104–115, 2006.

[13] M. M. Beigi and A. Zell. A novel kernel-based method for local pattern extraction in random process signals. In *15th European Symposium on Artificial Neural Networks, Bruges, Belgium (ESANN 2007)*, pages 265–270, April 2007.

[14] M. M. Beigi and A. Zell. Object Detection in Biosonar Based Robot Navigation. In *Worhshop on Planning, Perception and navigation fot Intelligent Vehicles, IEEE International Conference on Robotics and Autmation (ICRA2007)*, 2007.

[15] M. M. Beigi and A. Zell. Synthetic Protein Sequence Oversampling method for Classification and remote homology detection in imbalanced protein data. In *1st International Conference on Bioinformatics Research and Development (BIRD 2007), Berlin, Germany*, pages 263–277, 2007.

[16] M. M. Beigi and A. Zell. A Boosting Approach for Object Classification in Biosonar Based Robot Navigation. In *The 2008 IEEE International Conference on Robotics and Automation (ICRA 2008)*, 2008.

[17] M. M. Beigi and A. Zell. FIR-based Classifiers for Animal Behaviour Classification. In *The 2008 International Joint Conference on Neural Networks (IJCNN2008)*, 2008.

[18] M. M. Beigi and A. Zell. A kernel-based method for pattern extraction in random process signals. *J. Neurocomputing*, 2008.

[19] I. Belozertseva, T. Kos, P. Popik, W. Danysz, and A. Bespalov. Antidepressant–like effects of mGluR1 and mGluR5 antagonists in the rat forced swim and the mouse tail suspension tests. *J. Eur. Neuropsychopharmacol.*, 2006.

[20] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support Vector Clustering. *Journal of Machine Learning Research*, 2:125–137, 2002.

[21] E. Bettler, L. Oliveira, L. F. Campagne, F. E. Cohhen, and G. Vriend. Gpcrdb information system for g protein-coupled receptors. *J. Nucleic Acids Res.*, 31(1):294–297, 2003.

[22] M. Bhasin and G. P. S. Raghava. Gpcrpred: an svm-based method for prediction of families and subfamilies of g-protein coupled receptors. *J. Nucleaic Acids res.*, 32:383–389, 2004.

[23] L. Bo, L. Wang, and L. Jiao. Training support vector machines using greedy stage-wise algorithm. In *The 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, volume 3518, pages 632–638, 2005.

[24] F. Borsini and A Melt. Is the forced swimming test a suitable model for revealing antidepressant activity? *J. Psychopharmacology*, 94:147–160, 1988.

[25] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[26] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[27] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. In *Proceedings of the National Academy of Sciences*, volume 97, pages 262–267, 2000.

[28] P. Bühlmann and B. Yu. Boosting with the l_2 loss: regression and classification. *J. American Statistical Association*, 98:324–339, 2003.

[29] E. A. Stolz C. W. Anderson and S. Shamsunder. Multivariate autoregressive models for classification of spontaneous electroencephalogram during mental tasks. In *IEEE Trans Biomed Eng.*, volume 45, pages 277 – 286, 1998.

[30] J. Carter, I. Dubchak, and S. R. Holbrook. A computational approach to identify genes for functional RNAs in genomic sequences. *Nucleic Acids Research*, 29(19):3928–3938, 2001.

[31] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines, 2001. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence and Research*, 16:321–357, 2002.

[33] I. D. Coope and C. J. Price. Frame-based methods for unconstrained optimization. *J. optim. and applications*, 106:261–274, 2000.

[34] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–2879, 1995.

[35] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[36] J. Cryan and I. Lucki. Antidepressant-like effects mediated by 5-hydroxytryptamine2c receptors. *J. Pharmacol. Exp. Ther.*, 295:1120–1126, 2000.

[37] J. Cryan, A. Markou, and I. Lucki. Assessing antidepressant activity in rodents: recent developments and future needs. *Trends. Pharmacol. Sci.*, 23:238–245, 2002.

[38] F. Cucker and S. Smale. On the Mathematical Foundations of Learning. *Bulletin of Americam Mathematical Society*, 39(1):1–49, 2001.

[39] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *Proc. ICASSP*, 2007.

[40] G. Dai and D.-Y. Yeung. Boosting Kernel Discriminant Analysis and Its Application to Tissue Classification of Gene Expression Data. In *International Conference on Adaptive and Natural Computing Algorithms(ICANNGA 2005)*, volume 32, pages 53–58, 2005.

[41] S. Degroeve, B. De Baets, Y. Van de Peer, and P. Rouz. Feature subset selection for splice site prediction. *Bioinformatics*, 18:S75–S83, 2002.

[42] M. Detke, M. Rickels, and I. Lucki. Active behaviors in the rat forced swimming test differently produced by serotonergic and noradrenergic antidepressant. *J. Psychopharmacology*, 121:66–72, 1995.

[43] D. W. Elrod and K. C. Chou. A study on the correlation of g-protein coupled receptor types with amino acid composition. *J. Protein Eng.*, 15:713—-715, 2002.

[44] A. Lecuyer F. Lamarche F. Lotte, M. Congedo and B. Arnaldi. A review of classification algorithms for eeg-based brain-computer interfaces. *J Neural Eng.*, 4:R1–R13, 2007.

[45] Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[46] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[47] G. Fung, M. Dundar, J. Bi, and B. Rao. A Fast Iterative Algorithm for Fisher Discriminant using Heterogeneous Kernels. In *Proceedings of the 21th Inter- national Conference on Machine Learning (ICML)*, 2004.

[48] W. Gao and M. Hinders. Mobile robot sonar backscatter algorithm for automatically distinguishing walls, fences, and hedges. *The International Journal of Robotics Research*, 25(2):135–145, 2006.

[49] T. Gartner. Exponential and geometric kernels for graphs. In *NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data*, volume 5, pages 49–58, 2002.

[50] T. Gartner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop*, volume 2773, 2003.

[51] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[52] D. R. Griffin. *Echos of bats and men*. Anchor Books Doubleday Co., Garden City, NY, 1959.

[53] D. R. Griffin. *Listening in the Dark: Acoustic orientation of bats and men*. Comstock Publishing Associates, Ithaca, New York, 1986.

[54] J.-E. Grunwald, S. Schörnich, and L. Wiegrebe. Classification of natural textures in echolocation. In *Natl. Acad. Sci. USA*, volume 101, pages 5670–5674, 2004.

[55] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, pages 49–52, 1902.

[56] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer–Verlag, 2001.

[57] D. Haussler. Convolution kernels on discrete structures. Technical report, UCSC-CRL-99-10, University of California Santa Cruz, 1999.

[58] J. He, M. Sosokina, L. T. Watson, and J. W. Zwolak. Data distributed parallelism with dynamic task allocation for a global search algorithm. In *High Performance Computing Symposium*, pages 164–172, 2005.

[59] J. He, L. T. Watson, N. Ramakrishnan, C. A. Shaffer, A. Verstak, J. Jiang, K. Bae, and W. H. Tranter. Dynamic data structures for a direct search algorithm. *Journal of Computational Optimization and Applications*, 23:5–21, 2002.

[60] T. E. Herbert and M. Bouvier. Structural and functional aspects of g protein-coupled receptor oligomerization. *J. Biochem. Cell Biol.*, 76:1–11, 1998.

[61] U. Hoffmann, G. Garcia, J.-M. Vesin, K. Diserens, and T. Ebrahimi. Boosting Approach to P300 Detection with Application to Brain-Computer Interfaces. In *Proceedings of the 2nd International IEEE EMBS Conference on Neural Engineering*, pages 97—-100, Arlington, Va, USA, March 2005.

[62] N-J Huan and R. Palaniappan. Neural network classification of autoregressive features from electroencephalogram signals for brain computer interface design. *J. Neural Eng.*, 1(3):142 – 150, 2004.

[63] Y. Huang, J. Cai, and Y. D. Li. Classifying g protein coupled receptors with bagging classification tree. *Journal of Computationa Biology and Chemistry*, 28:275–280, 2004.

[64] J.-P.Vert, H. Saigo, and T.Akustu. Convolution and local alignment kernel. In B. Schölkopf, K. Tsuda, and J.-P.Vert, editors, *Kernel Methods in Compuatational Biology*. The MIT Press, 2004.

[65] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1-2):95–114, 2000.

[66] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, volume 11, 1999.

[67] T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*. Society for Artificial Intelligence in Statistics, 1999.

[68] N. Japkowicz. Learning from imbalanved data sets: A comparison of various strategies. In *Proceedings of Learning from Imbalanced Data*, pages 10–15, 2000.

[69] N. Japkowicz, C.Myers, and M. Gluch. A novelty detection approach to classification. In *Proceeding of the Fourteenth Int. Joint Conf. on ArtificialInteligence*, pages 10–15, 1995.

[70] F. Jiao, J. Xu, L. Yu, and D. Schuurmans. Protein fold recognition using the gradient boost algorithm. In *Computational Systems Bioinformatics Conference (CSB2006)*, 2006.

[71] T. Joachims. Making Large scale SVM learning practical. Technical report, LS8-24, Universitat Dortmund, 1998.

[72] T. Joachims. Making Large–Scale Support Vector Machine Learning Practical. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. USA. MIT Press.

[73] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79:157–181, 1993.

[74] R. Karchin, K. Karplus, and D. Haussler. Classifying g-protein coupled receptors with support vector machines. *J. Bioinformatics*, 18(1):147–159, 2002.

[75] H. Kashima and A. Inokuchi. Kernels for graph classification. In *ICDM Workshop on Active Mining: The 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 23–30, 2002.

[76] L. Kaufmann. Solving the Quadratic Programming Problem arising in Support Vector Classification. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 147–168, Cambridge, MA, 1999. USA. MIT Press.

[77] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. In *IEEE Transactions on Neural Networks*, volume 11, pages 124–136, 2000.

[78] S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46(1-3):351–360, 2002.

[79] J. Kim, E. N. Moriyama, C. G. Warr, P. J. Clyne, and J. R. Carlson. Identification of novel multi-transmembrane proteins from genomic databases using quasi-periodic structural properties. *J. Bioinformatics*, 16(9):767–775, 2000.

[80] G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *J. Math. Anal. Appli.*, 33:82–95, 1971.

[81] R. I. Kondor and J. Laerty. Diffusion kernels on graphs and other discrete input space. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[82] R. Kuc. Transforming echoes into pseudo-action potentials for classifying plants. *J. Acoust. Soc. AM.*, 110:2198–2206, 2001.

[83] D.N. Lee, F.R. van der Weel, T. Hitchcock, E. Matejowsky, and J.D. Pettigrew. Common Principles of Guidance by Echolocation and Vision. *Journal of Comparative Physiology*, 171(5):563–571, 1992.

[84] C. Leslie, E. Eskin, A. Cohen, J. Weston, and W.S. Noble. Mismatch string kernel for svm protein classification. *Advances in Neural Information Processing System*, pages 1441–1448, 2003.

[85] C. Leslie, E. Eskin, and W.S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, 2002.

[86] C. Leslie and R. Kuang. Fast Kernels for Inexact String Matching. *Journal of Machine Learning Research*, 5:1435–1455, 2001.

[87] N. Levinson. The wiener rms (root-means-squares) error criterion in filter design and prediction. *J. Math Phys.*, 25:261 –278, 1947.

[88] C.J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. In *IEEE Transactions on Neural Networks*, volume 13, pages 1045–1052, 2002.

[89] N. Littlestone. Learning Quickly when Irrelevant Attributes Abound: a New Linear–Threshold Algorithm . *Machine Learning*, 2(4):285–318, 1988.

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

[91] P. McKerrow and N. Harper. Plant acoustic density profile model of ctfm ultrasonic sensing. *J. IEEE Sensors*, 1(4):245–255, 2001.

[92] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48, 1999.

[93] P. Moser and D. Sanger. 5–HT1A receptor antagonist neither potentiate nor inhibit the effects of fluoxetine and befloxatone in forced swim test in rats. *Eur. J. Pharmacol.*, 372(12):127–134, 1999.

[94] C. F. Moss and A. Surlykke. Auditory scene analysis by echolocation in bats. *J. Acoust. Soc. Am.*, 110(4):2207–2226, 2001.

[95] S. Mukherjee, P. Tamayo, D. Slonim, A. Verri, T. Golub, J. Mesirov, and T. Poggio. Support vector machine classification of microarray data. Technical report, AI Memo 1677, Massachusetts Institute of Technology, 1999.

[96] S. Mukherjee, P. Tamayo, D. Slonim, A. Verri, T. Golub, J. Mesirov, and T. Poggio. Support vector machine classification of microarray data. Technical report, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2000.

[97] R. Müller. A computational theory for the classification of natural biosonar targets based on a spike code. *Network: Computat. Neural Syst. J.*, 14:595–612, 2003.

[98] V. P .S Naidu and M.R.S Reddy. Autoregressive (AR) based power spectral analysis of heart rate time series signal (HRTS signal). In *Conference on Convergent Technologies for Asia-Pacific Region (TENCON)*, volume 4, pages 1391 – 1394, 2003.

[99] E. Osuna, R. Freund, and F. Girosi. An Improved Training Algorithm for Support Vector Machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII –Proceedings of the 1997 IEEE Workshop*, pages 276–285, New York, USA, 1997. IEEE Press.

[100] M. Pazzini, C. Marz, P. Murphi, K. Ali, T.Hume, and C. Bruk. Reducing misclassification costs. In *proceedings of the Eleventh Int. Conf. on Machine Learning*, pages 217–225, 1994.

[101] J. Platt. A Fast Algorithm for Training Support Vector Machines. Technical report, MSR–TR–98–14, Microsoft Research, 1998.

[102] J. Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–205, Cambridge, MA, 1999. USA. MIT Press.

[103] T. Poggio and S. Smale. The Mathematics of Learning: Dealing with Data. *Notices of the American Mathematical Society (AMS)*, 50(5):537–544, 2003.

[104] R. D. Porsolt, G. Anton, N. Blavet, and M. Jalfre. despair in rats: A new model sensitive to antidepressant treatments. *Eur. J. Pharmacol*, 47:379–391, 1978.

[105] F. Provost and T. Fawcett. Robust classification for imprecise environments. *J. Machine Learning*, 423:203–231, 2001.

[106] B. Qian, O. S. Soyer, and R. R. Neubig. Depicting a protein's two faces: Gpcr classification by phylogenetic tree-based hmm. *FEBS Lett.*, 554(1-2):95–99, 2003.

[107] D. Ratner and P. McKerrow. Classification of natural textures in echolocation. In *Australian Conference on Robotics and Automation (ACRA)*, pages 104–110, 2001.

[108] A. Magnani S.-J. Kim and S. Boyd. Optimal Kernel Selection in Kernel Fisher Discriminant Analysis. In *roceeding of the 23th Int. Conf. on Machine Learning (ICML)*, pages 465–472, 2006.

[109] H. Saigo, J. P. Vert, N. Ueda, and T. akustu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[110] B.N. Schenkman. *Human Echolocation as a Function of Kind of Sound Source and Object Position.* Uppsala Psychological Reports, 1983.

[111] B.N. Schenkman. The Detection and Localization of Objects by the Blind with the Aid of Long-Cane Tapping Sounds. *Human Factors*, 28(5):607–618, 1986.

[112] B. Schölkopf, J. C. Platt, J. Shawe–Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High–Dimensional Distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[113] B. Schölkopf and A. J. Smola. *Learning with kernels–Support Vector Machines, Reglarization, Optimization, and Beyond.* MIT Press, Cambridge, 2002.

[114] J. Shawe –Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[115] S. Shevade, S. Keerthi, C. Bhattacharyya, and K. Murthy. Improvements to the SMO Algorithm for SVM Regression. In *IEEE Transactions on Neural Networks*, volume 11, pages 1188–1193, 2000.

[116] H. Shimodaira, K. I. Noma, M. Nakai, and S. Sagayama. Dynamic Time–Alignment Kernel in Support Vector. In *Advances in Neural Information Processing Systems*, volume 2, pages 921–928, 2001.

[117] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[118] A.J. Smola, P.J. Bartlett, and B. Schölkopf. *advances in large margin classifiers*. MIT Press, 2000.

[119] I. Steinwart. Sparseness of Support Vector Machines. *Journal Machine Learning Research*, 14:199–222, 2003.

[120] T. A. Stroffregen and J. B. Pittenger. Human echolocation as a basic form of perception and action. *Echological Psychology*, 7(3):181–216, 1995.

[121] J. Strum. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. `http://sedumi.mcmaster.ca/`.

[122] N. Suga. Biosonar and neural computation in bats. *Scientific American J.*, 262(6):60–68, 1990.

[123] P. Sun and X. Yao. Boosting Kernel Models for Regression. In *Sixth IEEE International Conference on Data Mining (ICDM'06)*, volume 35, pages 538–591, 2006.

[124] J. Swet. Measuring the accuracy of diagnostic systems. *J. Science*, 240:1285–1293, 1988.

[125] K. Takamori, S. Yoshida, and S. Okuyama. Effect of ACTH on the imipramine and desipramine–induced decrease in duration of immobility time as measured in a rat forced swimming test. *J. Life Sciences*, 69:1891–1896, 2001.

[126] D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.

[127] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustalw: improving the sesitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[128] K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *J. Bioinformatics*, 18:268S–275S, 2002.

[129] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[130] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[131] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on AI*, pages 55–60, 1999.

[132] J.-P. Vert and M. Kanehisa. Graph driven features extraction from microarray data using diffusion kernels and kernel cca. In *Advances in Neural Information Processing Systems*, volume 15, 2003.

[133] G. Walker. On periodicity in series of related terms. In *Proceedings of the R. Soc. of London. Series A.*, volume 131, pages 518–532, 1931.

[134] M. Wang. *Natural Landmark Classification with a Biosonar based Mobile Robot*. PhD thesis, University of Tübingen, Germany, 2006.

[135] M. Wang and A. Zell. Classification of natural landmarks with biosonar. *Journal of the Acoustic Society of America (JASA)*, 116:2640–2645, 2004.

[136] L. T. Watson and C. A. Baker. A fully distributed parallel global search algorithm. *Journal of Engineering Computations*, 18(1):514–579, 2001.

[137] G. Wu and E. Chang. Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 Workshop on Learning from Imbalanced Data Sets II,Washington, DC*, 2003.

[138] G. U. Yule. On a method of investigating prediction in disturbed seris. With special reference to Wölfer's sunspot numbers. *Philos. Trans. of Royal Soc. series A.*, pages 226 – 267, 1927.