IDIAP RESEARCH REPORT

PRIOR KNOWLEDGE IN KERNEL METHODS

Alexei Pozdnoukhov

IDIAP–RR 06-66

IDIAP Research Institute **www.idiap.ch**
Rue du Simplon 4 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 06-66

PRIOR KNOWLEDGE IN KERNEL METHODS

Alexei Pozdnoukhov

Abstract. This thesis explores approaches towards learning with kernel methods using prior knowledge. Invariant learning with kernel methods is considered in more details. In the first part of the thesis, kernels are developed which incorporate prior knowledge on invariant transformations. Next, algorithms which specifically include prior knowledge are considered. An algorithm which linearly classifies distributions by their domain was developed. In the last part of the thesis, the use of unlabelled data as a source of prior knowledge is considered. The technique of modelling the unlabelled data with a graph is taken as a baseline from semi-supervised manifold learning. For classification problems, we use this apporach for building graph models of invariant manifolds. For regression problems, we use unlabelled data to take into account the inner geometry of the input space.

Abstract

Machine Learning is a modern and actively developing field of computer science, devoted to extracting and estimating dependencies from empirical data. It combines such fields as statistics, optimization theory and artificial intelligence. In practical tasks, the general aim of Machine Learning is to construct algorithms able to generalize and predict in previously unseen situations based on some set of examples. Given some finite information, Machine Learning provides ways to exract knowledge, describe, explain and predict from data.

Kernel Methods are one of the most successful branches of Machine Learning. They allow applying linear algorithms with well-founded properties such as generalization ability, to non-linear real-life problems. Support Vector Machine is a well-known example of a kernel method, which has found a wide range of applications in data analysis nowadays.

In many practical applications, some additional prior knowledge is often available. This can be the knowledge about the data domain, invariant transformations, inner geometrical structures in data, some properties of the underlying process, etc. If used smartly, this information can provide significant improvement to any data processing algorithm. Thus, it is important to develop methods for incorporating prior knowledge into data-dependent models.

The main objective of this thesis is to investigate approaches towards learning with kernel methods using prior knowledge. Invariant learning with kernel methods is considered in more details.

In the first part of the thesis, kernels are developed which incorporate prior knowledge on invariant transformations. They apply when the desired transformation produce an object around every example, assuming that all points in the given object share the same class. Different types of objects, including hard geometrical objects and distributions are considered. These kernels were then applied for images classification with Support Vector Machines.

Next, algorithms which specifically include prior knowledge are considered. An algorithm which linearly classifies distributions by their domain was developed. It is constructed such that it allows to apply kernels to solve non-linear tasks. Thus, it combines the discriminative power of support vector machines and the well-developed framework of generative models. It can be applied to a number of real-life tasks which include data represented as distributions.

In the last part of the thesis, the use of unlabelled data as a source of prior knowledge is considered. The technique of modelling the unlabelled data with a graph is taken as a baseline from semi-supervised manifold learning. For classification problems, we use this apporach for building graph models of invariant manifolds. For regression problems, we use unlabelled data to take into account the inner geometry of the input space.

To conclude, in this thesis we developed a number of approaches for incorporating some prior knowledge into kernel methods. We proposed invariant kernels for existing algorithms, developed new algorithms and adapted a technique taken from semi-supervised learning for invariant learning. In all these cases, links with related state-of-the-art approaches were investigated. Several illustrative experiments were carried out on real data on optical character recognition, face image classification, brain-computer interfaces, and a number of benchmark and synthetic datasets.

Keywords: machine learning, kernel methods, support vector machine, invariances, semi-supervised learning, manifold regularization, unlabelled data

Version Abrégée

L'apprentissage automatique est un domaine actif de l'informatique moderne. Il s'agit d'extraire et d'estimer des dépendances à partir de données empiriques. Ce domaine est à l'intersection des statistiques, de la théorie de l'optimisation et de l'intelligence artificielle. Sur des tâches pratiques, le but général de l'apprentissage automatique est de construire, à partir d'un ensemble d'exemples, des algorithmes capables de généraliser, c'est-à-dire de prédire dans des situations inconnues auparavant. A partir d'une information finie, l'apprentissage automatique permet d'extraire des connaissances, de les décrire, d'expliquer et de prédire à partir de données connues.

Les méthodes à base de noyau forment l'une des branches les plus fructueuses de l'apprentissage automatique. Elles permettent d'appliquer des algorithmes linéaires, dont les propriétés sont bien connues, aux problèmes non-linéaires du monde réel. Les Machines à Vecteurs de Support (SVM) sont un exemple bien connu de méthodes à base de noyau. On les trouve aujourd'hui dans un large éventail d'applications d'analyse de données.

Dans de nombreuses applications pratiques, une connaissance *a priori* est disponible. Ce peut être une connaissance du domaine des données, de l'invariance par certaines transformations, de la structure géométrique interne aux données, de propriétés spécifiques aux processus sous-jacents, etc. Utilisées intelligemment, ces informations peuvent amener des améliorations significatives à n'importe quel algorithme de traitement des données. Il est donc important de développer des méthodes pour incorporer cette connaissance *a priori* dans les modèles dérivés des données.

L'objectif principal de cette thèse est d'utiliser la connaissance *a priori* dans les méthodes à base de noyau. Une attention particulière est portée à l'apprentissage d'invariance avec les méthodes à base de noyau.

La première partie de cette thèse propose des noyaux qui incorporent la connaissance *à priori* de transformations invariantes. Ces noyaux sont utilisables lorsque les transformations en question produisent un objet autour de chaque échantillon, en supposant que tous les points d'un tel objet appartiennent à la même classe. Différents types d'objets sont considérés, dont des objets géométriques "durs", et des distributions. Ces noyaux sont alors appliqués à la classification d'images avec les SVM.

Ensuite, des algorithmes qui incluent la connaissance *a priori* de fa con spécifique sont considérés. Un algorithme est proposé qui classifie, de fa con linéaire, les distributions suivant leur domaine. Il permet d'appliquer les méthodes à base de noyau pour résoudre des problèmes non-linéaires. De cette fa con, il combine le pouvoir discriminatif des SVM avec la structure bien développée des modèles génératifs. Il peut être appliqué à de nombreuses tâches réelles, dès lors qu'elles incluent des données représentées par des distributions.

La dernière partie de cette thèse considère l'usage de données sans label, comme source

de connaissance *a priori*. Comme méthode de base, nous nous inspirons de l'apprentissage semi-supervisé d'hypersurface, et modélisons les données sans label à l'aide d'un graphe. Pour les problèmes de classification, nous utilisons cette approche pour construire des modéles graphiques d'hypersurfaces invariantes. Pour les problèmes de régression, nous utilisons les données sans label pour prendre en compte la géométrie intrinsèque des données.

Pour conclure, dans cette thèse nous développons plusieurs approches pour incorporer une connaissance *a priori* dans les méthodes à base de noyau. Nous proposons des noyaux invariants pour des algorithmes existants, puis développons de nouveaux algorithmes. Enfin, nous adaptons à l'apprentissage d'invariance une technique utilisée en apprentissage semi-supervisé. Des expériences sont conduites sur des données réelles : reconnaissance optique de caractères, classification d'images faciales, interfaces cerveau-machine, ainsi que sur plusieurs tests de références et sur des données synthétiques.

Mots-clés: apprentissage automatique, méthodes à noyaux, machines a vecteurs de support, invariances, apprentissage de varietés et semi-supervisé, données non-labellées

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Machine Learning

Learning can be defined as a modification of a behavior through experience. “Learning to generalize in unseen situations” and not “learning by heart” is of our interest when we refer to this term in this study. The Machine Learning (ML) field aims to explore ways to make machines learn. It focuses on algorithms which can learn and disregards the hardware necessary to implement the algorithms.

Building a model from incomplete information in order to predict as accurately as possible some underlying structure of the unknown reality is considered to be the main task of ML. For this problem formulation to make sense to a mathematician, the information about the structure, the experience, and the model have to be expressed numerically. Thus, from the statistical point of view, the problem of learning becomes the one of function estimation from empirical measurements.

It is convenient to group learning problems into different domains. The particular properties of these domains generally give the need for the special properties of the types of functions used to describe the underlying reality. Indeed, considering solutions to problems in certain specific domains, one can construct learning machines with properties that enable them to solve tasks which would at present be impossible to solve in the general case. Thus, the philosophy of building a particular machine for a particular task which avoids solving general problems at an intermediate step (if possible) is accepted while building learning algorithms. The “Occam’s Razor” principle is another formulation of the latter.

Besides the property of a learning machine to be a good (precise) estimator of the unknown function, we would like it to be able to estimate the function in a practical amount of time. Thus, optimization theory comes into ML as an important part.

We will come back to these considerations throughout the thesis, when we will develop new approaches for learning from empirical data with prior knowledge. Both precision of the learning machine in terms of expected generalization performance and computational speed will be considered as the main features to improve.

1.2 Learning with Prior Knowledge

It is often desirable to include some prior knowledge to the data model. By prior knowledge we mean any information about the task that is given in addition to the training samples. Being one of the most successful learning machines, human can provide a valuable additional knowledge to implement into a learning algorithm.

In this general setting, the usual smoothness assumption (the smoothness of the modeling function in the vicinity of training samples) is also accepted based on a kind of prior knowledge. Consider the regression problem in Figure 1.1. The smoothness assumption is actually the prior information that the derivatives of the underlying function are small. Knowing the exact values of the derivatives in the training samples, the prediction could be improved even more. Thus, the additional knowledge about the derivatives provide better solutions.

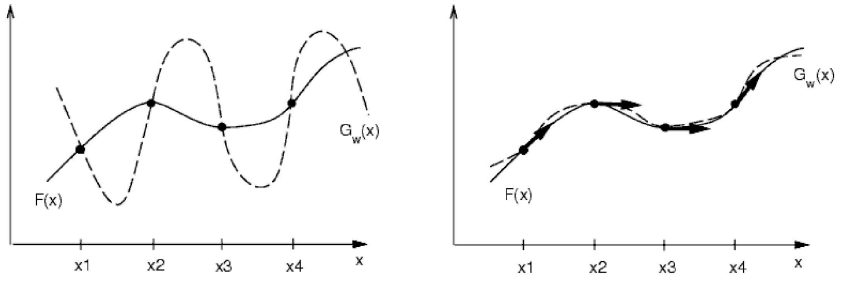


Figure 1.1. Prior Knowledge in Regression. Solid line represents the underlying dependency F(x). Left: dashed line represents one of the possible regression estimations G(x) based on training samples. Right: the additional knowledge on function derivatives allows for better prediction.

One of the most useful prior information is a knowledge on the invariances of the problem. We mean by the notion of invariance that the output of the learning system is unchanged while the inputs are transformed by some operator. Invariance under shifts and small rotations in the field of character recognition is a well-known example. Consider the example presented in Figure 1.2. Given the set of training samples of two different classes and a problem to classify the input space into two domains of the respective classes (Figure 1.2, center), it can be solved by a number of different ways. The two different partitions, shown in Figure 1.2, left and right, are facing two completely different solutions, however they both fit the data perfectly. The difference between them is a type of invariance they follow. Thus, the prior knowledge about invariances provides valuable additional information in this classification task. The invariance learning is a mainstream of the presented research.

Another important knowledge comes from unlabelled data. The information one obtains from it can be of different nature. A reasonable assumption to make is the following. Assume the data lies on some lower-dimensional manifold in the original input space. Using some properties of the manifold, data analysis methods can be improved. Particularly, unlabelled data, i.e. the samples without output information such as class labels or regression value, can reflect the structure of the underlying process. Considering Figure 1.3, let us illustrate the solution to a simple binary classification problem. In the left part of the

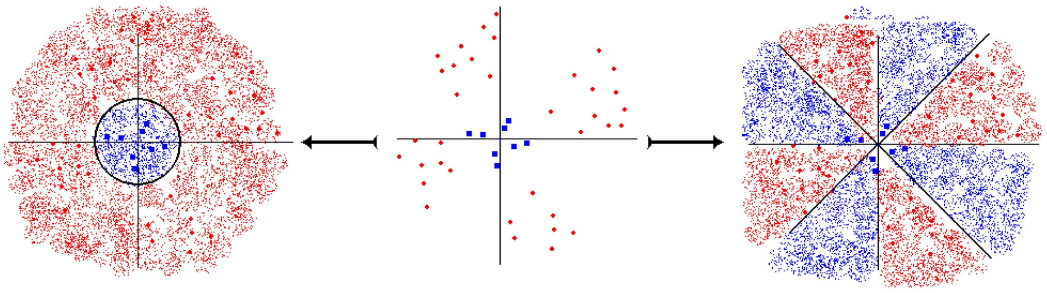


Figure 1.2. Prior Knowledge in Classification. Prior knowledge on rotational invariance (left) and scaling invariance (right) ask for completely different classification models for the same dataset (center).

figure the reasonable decision boundary is some kind of a line or any other smooth function which separates the samples of different classes. However, the understanding of the problem changes in the presence of unlabelled data (black dots in Figure 1.3, right). This information precises the structure of classes. Thus, the reasonable classification boundary changes significantly in the presence of unlabelled data.

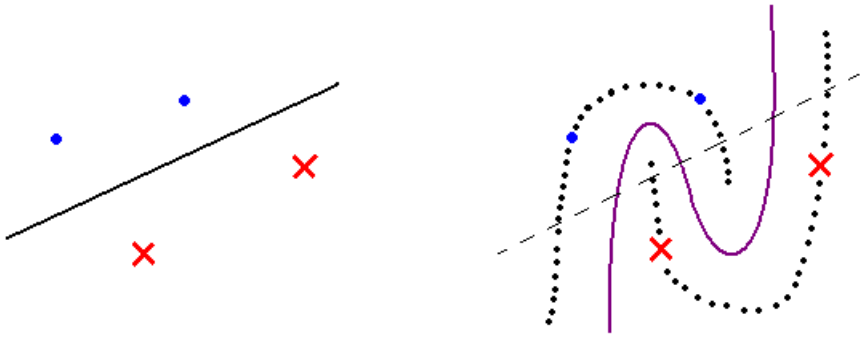


Figure 1.3. Classification in the presence of unlabelled data. The linear decision boundary (left) changes into non-linear to reflect the structure of classes, given unlabelled data (right).

Similar kinds of prior knowledge in regression, such as the geometry of the manifold the data belongs to, can improve the predictive performance of the learning machine.

These ideas will be elaborated in the presented research.

1.3 Challenges and Objectives

One of the most important fields in Machine Learning concerns the so-called Kernel Methods. This is a large group of methods which tackles most of the data analysis problems: classification, regression, density estimation, dimensionality reduction, etc. Many traditional algorithms, e.g. principal component analysis, discriminant analysis, ridge regression, etc. were "kernelised" to solve nonlinear problems in high dimensional spaces.

The most famous algorithm from the family of kernel methods is the Support Vector Machine (SVM) which is based on Statistical Learning Theory (SLT). At present SLT is

still under intensive development and SVMs find new areas of application. Being a data-driven approach, SVM is well suited to many data analysis tasks. SVMs develop robust and nonlinear data models with excellent generalisation abilities. SVMs are extremely good when input space is high dimensional and training data set is not big enough for traditional methods to develop a consistent nonlinear model.

Despite their extreme power of data modelling, SVM research and applications still have some room for challenging developments. Kernel methods provide solutions as an expansion of weighted kernel functions. An important question which arises in this framework is *the choice of the kernel function* as it explicitly defines the feature space hence it is of crucial meaning for the performance of the algorithms. The kernel function is also a factor that defines the capacity of the model. *Kernel design methodology*, and the particular problem of *incorporating some prior knowledge into the kernel function* is an important reseach direction. The same problems can be approached using the information from unlabelled data.

Combining the generalization abilities of SVMs (garanteed by the results of Statistical Learning Theory) and prior knowledge into an advanced kernel algorithm opens promising perspectives for further improvements in real-life use of kernel methods. The incorporation of prior knowledge on the *invariances* of the problem is of particular interest. The preferable directions are oriented to obtaining a general approach to the problem rather than in particular application-dependent solutions.

1.4 Contributions

The following main contributions of the thesis can be mentioned.

- **Invariant Kernels.** New kernels were developed that incorporate the knowledge about possible sample transformations into the learning algorithm. These kernels can be used in classification methods such as Support Vector Machines. They apply when the prior knowledge can be formalized by the description of an object around each sample of the training set, assuming that all points in the given object share the same desired class. A number of implementation techniques of this method, based on hard geometrical objects and soft objects based on distributions are considered. The method was applied to real-life task of face images classification and EEG signals classification. The results of this research were published in [55], [50].

- **Invariant Algorithms.** A new algorithm for classifying distributions by their domains was developed. The algorithm combines the principle of margin maximization and a kernel trick, applied to distributions. Thus, it combines the discriminative power of Support Vector Machines and the well-developed framework of generative models. It can be applied to a number of real-life tasks which include data represented as distributions. The algorithm can also be applied for introducing some prior knowledge on invariances into a discriminative model. This was applied to face image classification task [54]. Other applications such as object categorization were discussed in [53].

- **Invariant Manifolds in Classification.** Graph-based manifold modeling, used in semi-supervised learning, was adapted for invariant learning with kernel methods.

The approach is based on building a kernel function on the graph modeling the invariant transformation manifold. It provides a way for taking into account nearly arbitrary transformations of the input samples. The approach is verified experimentally on the task of optical character recognition in paper [51], providing state-of-the-art performance on harder problem settings.

- **Unlabelled Data in Regression.** A semi-supervised kernel method for regression estimation in the presence of unlabeled patterns was introduced. The method exploits a recently proposed data-dependent kernel which is constructed in order to represent the inner geometry of the data. This kernel was implemented into Kernel Regression methods. Experimental results revealed the properties of the method and its advantages as compared to fully supervised approaches. The results of this study are published in [52].

- **Links.** Several links between existing and newly developed approaches are described in the thesis. Regularization techniques, Vicinal Risk Minimization, Virtual Samples approaches are also described and compared to the developed methods.

1.5 Organization of the thesis

This thesis is organized as follows. First, we present some background of Machine Learning and the basics of the Statistical Learning Theory in Chapter 2. We then introduce the main learning algorithm which will be used in this study, the Support Vector Machine. The important notion of kernel functions is presented and plugged into the SVM framework. Next, in the same chapter, we outline the state-of-the-art techniques of invariant learning with kernel methods. We then present the contributions of the presented research. Chapter 3 presents a method for modifying the kernel function of SVM classifiers in order to include some invariant knowledge. Chapter 4 presents some developments devoted to constructing algorithms for classifying objects instead of samples. The particular method developed is a classifier of distributions according to the domain. The application of this approach are discussed. Next, in Chapter 5 we discuss the use of unlabelled data and the respective field of semi-supervised learning. We apply the techniques developed for semi-supervised learning to the problem of invariant learning in classification. We also apply these methods for high-dimensional regression estimation on the hidden low-dimensional manifolds. Chapter 6 concludes the thesis with discussions on current and future research directions of the presented ideas.

Chapter 2

Learning With Kernels

Kernel-based analysis is a very powerful tool for mathematicians, scientists and engineers. It provides a surprisingly rich and efficient way for data mining and pattern recognition and spreads from splines and Gaussian processes to neural networks [27], [58]. Kernel-based methods cover very different basic and applied scientific fields: statistics, biocomputing, finance, image analysis, geosciences, signal treatments, vision and speech recognition, and many others [27], [66]. Futhermore, the Statistical Learning Theory (SLT) (Vapnik-Chervonenkis theory [74]) establishes solid methodological and theoretical framework for many kernel-based algorithms.

The most famous algorithm from the family of kernel methods is the Support Vector Machine (SVM). At present SLT is still under intensive development and SVMs find new areas of application. Being a data-driven approach SVMs are well suited to many data analysis tasks. They develop robust and nonlinear data models with excellent generalisation abilities. Moreover, they are extremely good when input space is high dimensional and training data set is not big enough for traditional methods to develop a consistent nonlinear model. SVMs use only support vectors (most important samples of the dataset) to derive decision boundaries between classes. In fact, Support Vector Machines have demonstrated one of the best results on different classification problems, ranging from bio-computing to handwritten digits recognition [22]. Despite their extreme power of data modelling, SVM research and applications still have some room for challenging developments: task dependent adaptation of kernels, incorporation of prior information into kernel-based models and hyper parameter tuning, task-specific kernel design, etc.

Many traditionally linear algorithms, e.g. principal component analysis, ridge regression, were "kernelised" to solve nonlinear problems in a high dimensional feature spaces. Numerous real case studies in different fields have demonstrated high efficiency of kernel-based methods in solving basic data analysis problems: data clustering, classification and pattern recognition, regression, distribution modelling, outliers and novelty detection and others, see for example [63] and references therein.

In this chapter, the basics of Learning with Kernels are presented, starting from fundamental definitions of Statistical Learning Theory to state-of-the-art in invariant learning with kernels.

2.1 Statistical Learning Theory

In the framework of statistical learning, by the term " *learning*" we mean estimating some function $y = f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^N$ and depending on the type of problem, $y \in \mathbb{R}$ for regression, $y \in [1, 2, ...M]$ for M-class pattern recognition or $y \in \{-1, 1\}$ for binary pattern recognition. The estimate has to be made given only examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)\}$ of the mapping performed by the unknown function. Practically, the process of learning is considered as follows. A learning machine must choose from a given set of functions $\{\mathbf{F} = f(\boldsymbol{x}, \alpha), \alpha \in \Lambda\}$ the one which best (in some predefined sense) approximates the unknown dependency. Λ is an abstract set of parameters, chosen beforehand. This choice is actually an optimization in the parameter space of α.

The constraints in the considered setting are quite general, providing robust framework within which to construct powerful learning algorithms. A probabilistic interpretation of the data is assumed. An empirical example is considered to be drawn according to some fixed but unknown distribution $P(\boldsymbol{x}, y)$, each point possibly being skewed by noise which also comes from a fixed but unknown distribution. The examples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell$ are assumed to be vectors in $\mathbb{R}^N$ and as such we are only permitted to solve problems that have some geometric interpretation. It restricts us from direct applications such as classification of strings of different lengths, raw signal classification of unknown duration, etc. In these cases, some preliminary preprocessing procedures have to be carried out. However, the presented setting includes many real-world applications in different fields.

Learning Algorithm. A number of notions have to be formalized to construct a learning algorithm. First, we need to define a learning problem with associated loss function, an induction principle, a set of decision functions, and finally an algorithm that implements these components. Let us first consider the learning problems which we are interested in, together with their associated loss functions.

Loss functions and Risk Minimization. We define learning as estimating the function $f(\boldsymbol{x})$ from the set of functions $\{\mathbf{F} = f(\boldsymbol{x}, \alpha \in \Lambda)\}$ defined a priori, which provides the minimum value of the risk function

$$R(\alpha) = \int Q(y, f(\boldsymbol{x}, \alpha)) dP(\boldsymbol{x}, y) \tag{2.1}$$

where $Q(y, f(\boldsymbol{x}, \alpha))$ is the loss function - a measure of discrepancy between the estimate and the actual value y given by the unknown function at a point $\boldsymbol{x}$. By defining our goal as minimizing the risk function we state that our objective is to minimize the expected average loss (as defined by the loss function we choose) for a given problem.

For this definition to be applied to real-life tasks, the types of learning problems with the associated loss functions have to be defined. In this work we mainly consider two basic learning problems (classification and regression), and mention the third one (density estimation). We describe them in the next section.

2.1.1 Three Main Learning Problems

In the framework of SLT [74] three main learning problems of function estimation are defined. The problems are: pattern recognition, regression estimation and density estimation. In this thesis we consider the first two, and mention density estimation briefly. In this

introduction we follow the traditional way for introducing Machine Learning framework, and review the definition of these problems in turn.

Pattern Recognition. In the problem of pattern recognition, each vector $\boldsymbol{x}$ can be labeled with one of two classes, i.e. the output $y \in \{-1, 1\}$. The loss function for this task measures the number of incorrectly classified patterns

$$Q(f, \boldsymbol{x}) = \begin{cases} 0, & if\ f(\boldsymbol{x}) = y, \\ 1, & otherwise. \end{cases} \tag{2.2}$$

For this loss, the risk (2.1) which is minimized measures the probability of classification error.

Regression Estimation. In the problem of regression estimation, the value of y at any point $\boldsymbol{x}$ is a real value, i.e. outputs $y \in \mathbb{R}$. In the general case of estimation any point $\boldsymbol{x}_i$ is measured with noise generated from a usually unknown distribution. It is known that if the regression function $f(\boldsymbol{x})$ belongs to the set $\mathbf{F} = \{f(\boldsymbol{x}, \alpha), \alpha \in \Lambda\}$, and noise distribution in the outputs is Gaussian, then it can be found by minimizing (2.1) with the squared loss function

$$Q(f, \boldsymbol{x}) = (f(\boldsymbol{x}) - y)^2. \tag{2.3}$$

Considering the links with approximation theory, where the data is noiseless, the choice of loss function depends on the metric one considers. That is, supposing the function $f(\boldsymbol{x})$ does not belong to the set $\mathbf{F}$, then we have to find the closest function to the unknown one in a given metric. Regularization framework [71] is introduced to solve such kinds of ill-posed problems.

Density Estimation. In the problem of density estimation one has to estimate the density function $p(\boldsymbol{x})$ of a random variable X given the i.i.d. data $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell\}$. For this problem, if the unknown density belongs to the set of densities $\{P(\boldsymbol{x}, \alpha), \alpha \in \Lambda\}$ it can be found using the following loss function

$$Q(\boldsymbol{x}, p) = -\log p(\boldsymbol{x}) \tag{2.4}$$

for the risk functional

$$R(\alpha) = \int Q(\boldsymbol{x}, p) dF(\boldsymbol{x}), \tag{2.5}$$

where $F(\boldsymbol{x})$ is the c.p.d.f. of X. The general solution of the density estimation problem is out of scope of this thesis, however, some referencies will be provided throughout the text.

2.1.2 Induction Principles and VC-dimension

An induction principle provides a method for generalizing some particular observations into a general rule. For example, it allows one to construct a decision rule that can classify every point in a space given only finite examples (points) from the space (the training set). First we will consider one of the simplest induction principles, the so-called Empirical Risk Minimization (ERM) principle. We will then review the main induction principle in statistical learning theory, the Structural Risk Minimization (SRM) principle, and discuss the similarities between SRM and Regularization theory. The Vicinal Risk Minimization principle is considered later as well.

The Empirical Risk Minimization principle

While minimizing the risk functional (2.1) we choose the function that provides the minimum deviation (in the sense of our loss function) from the true function across the whole function space for every point $\boldsymbol{x}$. In reality, however, the joint distribution function $P(\boldsymbol{x}, y)$ is unknown, and we do not have the value of y for each point $\boldsymbol{x}$ in the function space, but only the training set pairs $\{\boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_\ell, y_\ell\}$. We can instead approximate function (2.1) by considering the following so-called empirical risk functional:

$$R_{emp} = \frac{1}{\ell} \sum_{i=1}^{\ell} Q(y_i, f(\boldsymbol{x}_i, \alpha)). \tag{2.6}$$

Then, a function that gives minimum to the empirical risk is chosen as an optimal decision (regression) function. This induction principle is called Empirical Risk Minimization. By this we only choose a decision rule based on its empirical performance on the finite number of known examples. One can calculate the value of $R_{emp}(\alpha)$ and hence select a choice of α which minimizes this value. However, the minimum of (2.6) does not necessarily approximate well the minimum of (2.1) when ℓ is small. We consider two basic improvements to this. First, note that the probability distribution which actually appears here (refer to Eq. (2.1)) is an empirical distribution $p(\boldsymbol{x}, y) = \frac{1}{\ell} \sum_{i=1}^{\ell} \delta(\boldsymbol{x} - \boldsymbol{x}_i)\delta(y - y_i)$. Later, more sophisticated distributions will be used which result in Vicinal Risk Minimization principle.

Now, we consider another extention of the ERM induction principle. The following result has actually gave rise to the explosive growth of Machine Learning methods. Let us consider the case of pattern recognition. For this problem, the following bound on expected risk holds with probability $1 - \eta$ [74].

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(\frac{2\ell}{h}) + 1) - \log(\frac{\eta}{4})}{\ell}}. \tag{2.7}$$

The parameter h, introduced here, is the VC dimension of the set of decision functions parameterized by α. For the case of classification, the VC dimension of a set of functions is the maximum number of points that can be separated in all possible ways by that set (see [75], p. 76 for details). Knowning the exact value of h and choosing a sufficiently small η one can use this bound to calculate the best choice of α (the best function to select from the set of decision functions). The ERM principle, which minimizes the empirical risk (the first term only) gives a small value of expected risk when $\frac{\ell}{h}$ (the ratio of the number of training samples to the VC dimension of the set of functions) is large. If the VC dimension of the set of functions is large the second term (the so-called confidence interval) will be large. To minimize over both terms the VC dimension of the set of functions would have to be a controlling variable and not just be chosen a priori. This is discussed in details in the next section.

VC Dimension and Generalization

So far, the aim in learning is to choose the function (decision rule) from the set of possible decision rules which best describes the data and the underlying process as well. Because

the amount of data is usually a small sample the function chosen may describe dependencies in the data but not in the unknown function as a whole. We call the ability to describe the actual underlying functional dependency from finite empirical data the *generalization ability* of a learning machine.

Generalization ability is controlled by choosing an appropriate set of functions $\mathbf{F} = \{f(\boldsymbol{x}, \alpha)\}$. The capacity of this set of functions, one measure of which is VC dimension, controls the empirical risk achieved. In classification, it is the number of possible separations of the data with the functions from this set. Choosing a set of functions which can perform many possible separations will achieve a low empirical risk but can generalize poorly. This phenomenon of choosing the false (too complex) structure is called overfitting. Choosing a low capacity, i.e. a weaker set of functions, can result in better generalization ability, but then the set of functions may be too weak to describe the necessary dependencies in the data. For example, at one extreme $\mathbf{F} = \{f(\boldsymbol{x}, \alpha)\}$ could be the set of linear decision functions (hyperplanes), at the other the set of sine functions. The former can only describe linear dependencies, whereas the latter is a non-falsifiable learning machine - it can describe any dependency with a high frequency sine curve and so generalization will not take place [75]. The generalization ability can be controlled by choosing the VC dimension or some other embodiment of capacity in the set of functions. In the next section, we describe the Structural Risk Minimization (SRM) induction principle which attempts to control both the empirical risk on the training data and the capacity of the set of functions used to obtain this value of risk.

The Structural Risk Minimization Principle.

For pattern recognition, the SRM principle is justified by the result (2.7). Its objective is to minimize both the empirical risk and the confidence interval - both terms in the bound (2.7). This can be thought as follows. Let us define a structure

$$S_1 \subset S_2 \subset \cdots \subset S_T \tag{2.8}$$

on the set of decision functions $\mathbf{F}$ whose VC dimensions satisfy

$$h_1 \leq h_2 \leq \cdots \leq h_T \tag{2.9}$$

and then choose an appropriate element S_{opt} of the structure that minimizes the bound (2.7). This can be approached by minimizing the empirical risk using each learning machine (each element S_i from the set) and selecting the one with lowest value of the bound. Thus the SRM principle defines a trade-off between the accuracy (empirical risk or training error) and complexity of the approximation by minimizing over both terms in (2.7). The theory of bounds is an important part of Machine Learning, since it provides the foundation for a new algorithm to possess good generalization abilities. Similar bounds for risk were derived (and are constantly improved) in SLT for other learning problems as well.

Vicinal Risk Minimization Principle.

This induction principle is an extention of ERM, with a more sophisticated density estimate rather then empirical density. It was introduced by Vapnik (2000) by considering local

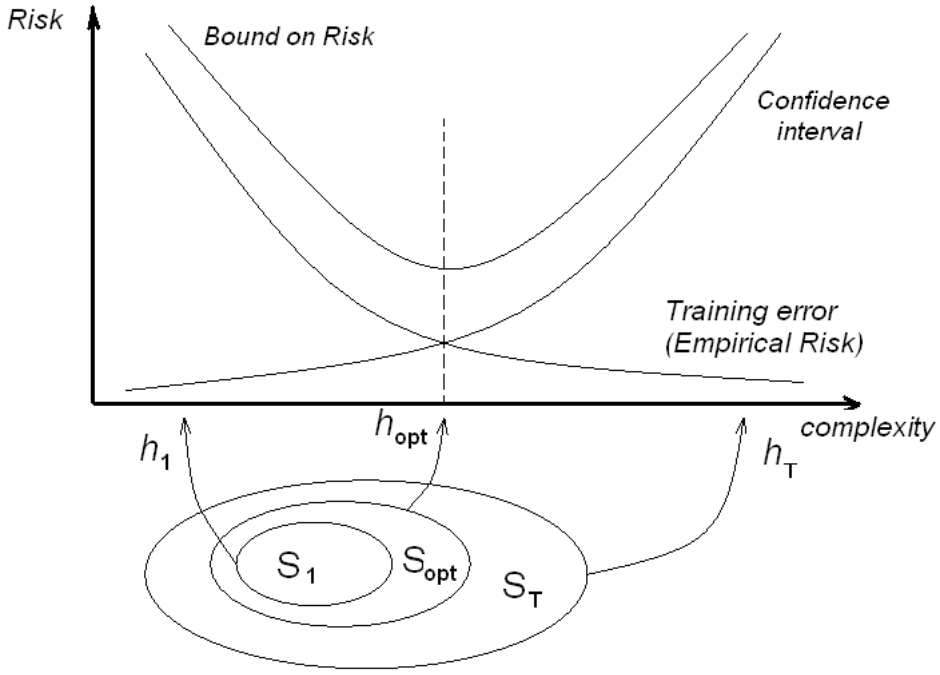


Figure 2.1. The bound on the risk is controlled by a trade-off between the empirical risk (training error) and the confidence interval (capacity of the set of functions).

distributions instead of delta functions for every sample. Defining the *vicinities* of the training samples (the support of the local distribution functions) and assuming some local density $p(\boldsymbol{x}|\boldsymbol{x}_i, r_i)$ therein, one obtains the following Vicinal Risk functional:

$$R_{vic}(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell} L\left(y - \int f(\boldsymbol{x},\alpha)p(\boldsymbol{x}|\boldsymbol{x}_i, r_i)d\boldsymbol{x}\right), \tag{2.10}$$

where $\boldsymbol{x}_i$ is a training sample and r_i is its vicinity parameter. Minimizing (2.10) instead of empirical risk is called the Vicinal Risk Minimization (VRM) principle.

Note that a number of learning algorithms can be achieved as a direct development of the VRM, considering the corresponding local density estimates. For example, as it was shown in [10], Ridge Regression, Logistic Classifiers, and some other methods can be derived from VRM. We consider these relations in more details later.

Vapnik mentions how to use the VRM principle to incorporate an invariance into the learning algorithm. Using the density functions $p(\boldsymbol{x}|\boldsymbol{x}_i, r_i)$ defined on the non-symmetrical support that describes the invariance to the desired transformation, one enforces the learning algorithm to obey the invariance's properties. This approach has definite links with regularization and invariant learning framework, discussed below.

2.2 Support Vector Learning

The particular learning machine we are interested in this work, the Support Vector (SV) Machine, implements the set of linear decision functions and uses the SRM principle. In

this section we will review how SV Machines are constructed for pattern recognition and regression estimation, and see how the available decision rules provide a general method of function estimation that is performed by solving only a convex (quadratic) optimization problem. In the following chapters we will then review and explore approaches for incorporating prior knowledge to the Support Vector learning. The practical aspects of the methods are discussed. The presented developments provide further extentions to the powerful, elegant and generally applicable Support Vector learning algorithms.

The Set of Decision Functions. We shall consider the following set of functions to construct the basic linear Support Vector learning machine:

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b, \tag{2.11}$$

where $\boldsymbol{x}$ is a vector in $\mathbb{R}^N$ (input space), b is a scalar, and $\boldsymbol{w}$ is an unknown vector in $\mathbb{R}^N$ to be optimized. For the case of classification, the sign of the function $f(\boldsymbol{x})$ is considered as a classifier output.

2.2.1 Support Vector Classification

SVMs are originally a kind of linear classifier. Let us consider the following decision function, defined by $\boldsymbol{w}$ and b:

$$y = \begin{cases} 1, \ if \ \boldsymbol{w} \cdot \boldsymbol{x} - b \geq 1 \\ -1, \ if \ \boldsymbol{w} \cdot \boldsymbol{x} - b \leq -1. \end{cases} \tag{2.12}$$

The difference with (2.11) is that now the decision is taken according to the position of the sample with respect to some margin along the hyperplane defined by $\boldsymbol{w}$. This is an important property, since the following result holds:

Lemma. If the training set of vectors in $\mathbb{R}^N$ belongs to the sphere with a radius R, the VC-dimension h of the set (2.12) is bounded with:

$$h \leq \min(\left[R^2 \|\boldsymbol{w}\|^2\right], N) + 1. \tag{2.13}$$

Hence the margin can be maximized to minimize the upper bound for VC dimension (2.13) and, correspondingly, the bound for the risk (2.7). This is the key idea of the Support Vector Machine classifier.

Thus, SVMs not only aim at separating two classes (as does the Perceptron algorithm, for example) but also at maximizing the margin between these two classes, as depicted in Figure 2.2. The intuitive idea is that a hyperplane with a large margin should be more resistant to noise than a hyperplane with a small margin. SVMs are thus often referred to as large margin classifiers. More formally, we first define strict separating constraints of the classes as

$$\begin{array}{l} \boldsymbol{w} \cdot \boldsymbol{x}_i + b, \ \text{if} \ y_i = 1 \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b, \ \text{if} \ y_i = -1. \end{array} \tag{2.14}$$

The scaling of $\boldsymbol{w}$ and b is arbitrary and fixed as above. For convenience sake, the constraints can be rewritten as

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \tag{2.15}$$

The margin ρ can be easily computed as the distance between the hyperplane $f(\boldsymbol{x}) = 1$ and the hyperplane $f(\boldsymbol{x}) = -1$, refer to the Figure 2.2.

$$\rho = \frac{2}{\|\boldsymbol{w}\|}. \tag{2.16}$$

Hence, the SVM algorithm has to maximize the margin (2.16) while respecting the con-

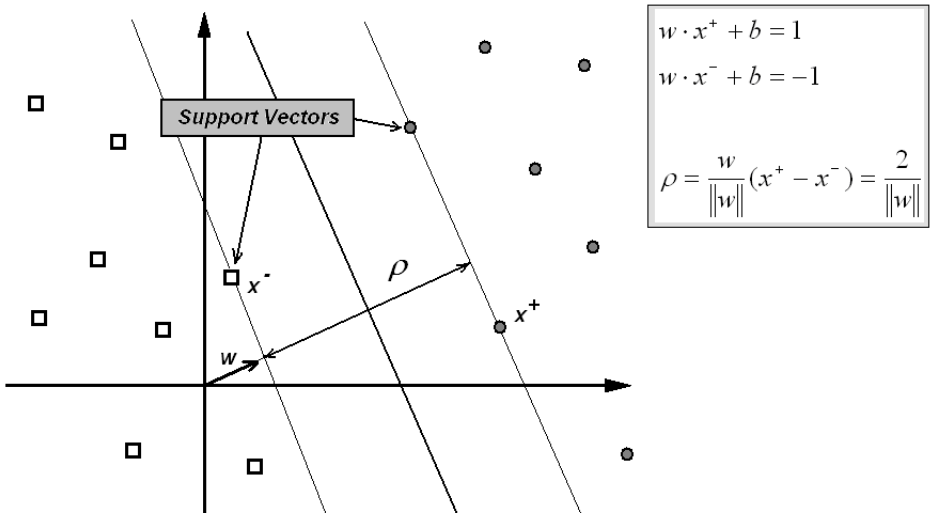


Figure 2.2. Maximizing the margin between classes leads to generalization in classification.

straints (2.15). This is usually done by minimizing the squared norm $\|\boldsymbol{w}\|^2$. This optimisation problem can be solved in Lagrangian formulation. Introducing the Lagrange multipliers α_i for the constraints, one has to minimize the Lagrangian $\mathcal{L}$ with respect to $\boldsymbol{w}$ and b and maximize with respect to α_i:

$$\mathcal{L}_P = \tfrac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i y(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) + \sum_{i=1}^{\ell} \alpha_i, \tag{2.17}$$

$$\alpha_i \geq 0, \ \ \forall i. \tag{2.18}$$

We obtain the saddle point, where derivatives of $\mathcal{L}_P$ with respect to the primal variables $\boldsymbol{w}$, b vanish:

$$\frac{\partial \mathcal{L}_P(\boldsymbol{w}, b, \alpha)}{\partial b} = 0, \quad \frac{\partial \mathcal{L}_P(\boldsymbol{w}, b, \alpha)}{\partial \boldsymbol{w}} = 0, \tag{2.19}$$

i.e.:

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0, \quad \sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{x}_i = 0. \tag{2.20}$$

These can be substituted to Lagrangian (2.17) to get the dual formulation of the problem:

$$\mathcal{L}_D = \sum_{i=1}^{\ell} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{2.21}$$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0, \quad \sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{x}_i = 0. \tag{2.22}$$

This formulation is the most widely used for the optimisation problem solving. It should be noted that the solution depends only on the dot product of the data with nonzero weights/coefficients, since the decision function becomes:

$$f(\boldsymbol{x}) = \sum_{i=1}^{\ell} y_i \alpha_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b. \tag{2.23}$$

The problem is a convex QP-problem with respect to linear constraints, and, in general, any of well-known methods can be applied to solve it [45], [73]. But there are some particular features that can be taken into account when developing QP-solvers [47].

From the Kuhn-Tucker conditions we obtain the following result. If $\alpha_i = 0$, then $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$, and for $\alpha_i > 0$, the equality holds: $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1$. These two possibilities, $\alpha_i = 0$ and $\alpha_i > 0$ give the name for the whole Support Vector method. The samples from training data corresponding to $\alpha_i > 0$ will fall on the hyperplanes $f(\boldsymbol{x}, \{\boldsymbol{w}, b\}) = +1$ or $f(\boldsymbol{x}, \{\boldsymbol{w}, b\} = -1$ of the decision surface. They are called the *Support Vectors*. Notice, that if we remove all other points except the SV from the training data set and train SVM on the SVs only, we obtain the same decision boundary, i.e. SVs have the determinant meaning for the given classification task. In particular, it gives us an opportunity to use the number of the SVs and their location as one of the criteria for the search of optimal SVM parameters.

Soft Margin Classifier. All the techniques developed up to now for linearly separable sets can be extended to the non-separable sets by adding slack variables $\xi_{\mathrm{i}} \geq 0$ to the constraints (2.15) [Cortes and Vapnik, 1995]:

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, \;\; i = 1, ..., L. \tag{2.24}$$

As few ξ_{i} as possible should be non zero, so now the task is to minimize the functional:

$$\tau(w, \xi) = \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell} \xi_i \tag{2.25}$$

subject to the constraints (2.24). The first term in (2.25) corresponds to minimising of the VC-dimension, the second one corresponds to minimising the number of misclassified points of the training set. The positive constant C is weighting the second criterion with respect to the first one. It also becomes an upper bound for the weights, resulting in the box-type constraints $C \geq \alpha_i \geq 0, \forall i$. This problem is also a QP-problem and it can be solved by standard QP-solvers.

In this form, the described Support Vector learning machine is of intensive use for data classification problems. Strictly speaking, the presented method is often referred to as Large Margin Classifier, and its *kernelised* version is now called the Support Vector Machine. This important kernel extention of the classifier will be discussed especially in Section 2.3.

2.2.2 Support Vector Regression

In the problem of regression estimation we are given a set of observations $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)\}$ generated from an unknown probability distribution $P(\mathbf{X}, \mathbf{Y})$ with $\boldsymbol{x}_i \in \mathbb{R}^{\mathrm{N}}, y_{\mathrm{i}} \in \mathbb{R}$ and a class of functions $\mathbf{F} = \{f|\mathbb{R}^{\mathrm{n}} \to \mathbb{R}, \alpha \in \Lambda\}$. Our task is to find a function f from the given class of functions that minimizes a risk functional (2.1). When it is known that the measurements are corrupted with additive normal noise, then minimization of the empirical risk with a quadratic loss function (2.3) results in a best unbiased estimator of the regression f in the selected class $\mathbf{F}$. But when it is only known that noise generating distribution is symmetric, the use of linear loss function is preferable, and results in a model from the so-called robust regression family. The Support Vector Regression model is based on the ε-insensitive loss functions. For example the linear ε-insensitive loss is defined as

$$Q(y, f(\boldsymbol{x})) = \begin{cases} |y - f(\boldsymbol{x})| - \varepsilon & \text{if } |y - f(\boldsymbol{x})| > \varepsilon \\ 0 & \text{otherwise.} \end{cases} \tag{2.26}$$

Following the Structural Risk Minimization principle, the model complexity has to be penalized simultaneously with keeping empirical risk (training error) small. In analogy to the classification case, the complexity of linear regression functions $\mathbf{F} = \{f(\boldsymbol{x})|f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b\}$ can be controlled by the term $\|\boldsymbol{w}\|^2$. We refer to [75] for the strict foundation of the latter. Introducing the slack variables ξ and the corresponding trade-off constant C, this results in the following optimization problem:

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \tag{2.27}$$

$$\text{subject to} \begin{cases} f(\boldsymbol{x}_i) - y_i - \varepsilon \leq \xi_i \\ -f(\boldsymbol{x}_i) + y_i - \varepsilon \leq \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \text{ for } i = 1, \ldots, \ell\ . \end{cases}$$

Introducing Lagrange multipliers leads to the following dual formulation of the problem:

$$\begin{aligned} \text{maximize} \quad & -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\boldsymbol{x}_i \cdot \boldsymbol{x}_j \\ & -\varepsilon\sum_{i=1}^{\ell}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{\ell} y_i(\alpha_i^* - \alpha_i) \end{aligned} \tag{2.28}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i) = 0 \\ 0 \leq \alpha_i^*, \alpha_i \leq C \text{ for } i = 1, \ldots, \ell\ . \end{cases}$$

This problem is a Quadratic Programming problem hence can be numerically solved by a number of standard or specialized methods ([47], [70], [73]). The prediction is a linear regression function:

$$f(\boldsymbol{x}) = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)\boldsymbol{x}_i \cdot \boldsymbol{x} + b \tag{2.29}$$

where b can be found easily given the constraints in 2.28. The same properties as for SV classification holds for SVR: the solution is sparse (usually, most of the α weights are zero), robust, and unique due to QP optimization. The hyper-parameters of SVR are therefore C and ε. The positive constant C is the parameter that defines the trade-off between training error and model complexity. In dual formulation C defines the upper bound of the multipliers α_i and α_i^* (2.28), hence defines the maximal influence a sample can have on the solution. This means that the more noisy the data the less should be the value of C. The positive constant ε is the width of the insensitive region of the loss function. This is the parameter that mainly defines the sparseness of the SVR solution - the points that lie inside the ε-tube have zero weights.

2.3 Kernels

Kernel Methods are playing an important role in Machine Learning being one of the theoretically well-founded methods which showed promising performances on real-life problems [63]. Support Vector Machines, one of the basic and most advanced algorithms, is a natural field for applying kernels. Given a training set $((\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ...(\boldsymbol{x}_\ell, y_\ell))$ of ℓ samples, most kernel methods typically model them with the following hypothesis function:

$$f(\boldsymbol{x}) = \sum_{i=1}^{\ell} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b, \tag{2.30}$$

where $K(\boldsymbol{x}_i, \boldsymbol{x})$ is a *kernel function* and α_i and b are the parameters to optimize. For binary classification task ($y_i \in \{+1, -1\}$) the decision is usually taken according to the sign of (2.30), and the weights α appear to be positive or negative according to the label of the corresponding sample.

The application field of kernel algorithms (basically, SVMs) is quite wide. It includes a number of tasks in the field of text processing [30], image processing [48], gene structure analysis [23], time series analysis and prediction [44], environmental applications [32].

2.3.1 Kernel Trick

The following result justifies the so called *k*ernel trick:

Theorem(Mercer). Consider a continuous symmetric function $K(\boldsymbol{x}, \boldsymbol{x}') : X^2 \to \mathbb{R}$ where we denoted an input space as X. If for any function $g \in \mathcal{L}_2(C)$, C is the compact subset of X,

$$\int_C \int_C K(\boldsymbol{x}, \boldsymbol{x}') g(\boldsymbol{x}) g(\boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}' \geq 0 \tag{2.31}$$

then it can be expanded in a absolutely and uniformly converging series

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{\infty} a_k \psi_k(\boldsymbol{x}) \psi_k(\boldsymbol{x}') \tag{2.32}$$

where $\psi_k(.)$ and $a_k \geq 0$ is the eigensystem of the corresponding integral operator with a kernel $K(\boldsymbol{x}, \boldsymbol{x}')$.

For us it means that for every function $K(\boldsymbol{x},\boldsymbol{x}')$ satisfying the conditions of the theorem there exists a feature space where it acts as a dot product. Mercer theorem gives one way of obtaining a dot product from a kernel function. Note that the exact mapping $\boldsymbol{x} \mapsto \{\sqrt{a_k}\psi_k(\boldsymbol{x})\}$ from input space to the feature space is undefined, but we can be sure that this space exists. If one wants to use a definite feature space and can provide the explicit mapping to it then the kernel function would be just a dot product in this feature space.

Therefore, given a kernel function and some learning algorithm formulated only in terms of the dot products between input samples, one easily obtains a non-linear form of the algorithm. This is the case for a number of contemporary algorithms such as Support Vector Machines both for classification [6] and regression [75], Kernel Ridge Regression [59],[49], Kernel Principle Component Analysis [61], Kernel Fisher Discriminant Analysis [41], etc.

Considering the optimization problems and the decision rules of Support Vector algorithms ((2.21), (2.23) for classification and (2.28), (2.29) for regression), one can see that the training samples only appear in the terms of dot products. Therefore, the kernel trick can be directly applied with substituting the dot products with kernels:

$$\boldsymbol{x} \cdot \boldsymbol{x}' \quad \mapsto \quad K(\boldsymbol{x},\boldsymbol{x}'). \tag{2.33}$$

Thus, a non-linear extension of the Support Vectror algorithms is derived. The basic model which is used in Support Vector learning is a kernel expansion (2.30).

2.3.2 Choosing Kernel Parameters

An important question which arises in this framework is thus *the choice of the kernel function*. Kernel functions explicitly define the feature space hence they are of crucial meaning for the performance of the algorithm. The kernel function is also a factor that defines the capacity of the model. *Kernel design methodology*, and the particular problem of *incorporating some prior knowledge into the kernel function* is an important part of the presented work.

There are several widely used kernels in SV learning. These are

- Polynomial kernel: $K(\boldsymbol{x},\boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}' + 1)^p$, $p \in \mathbb{N}$.
- Gaussian RBF kernel: $K(\boldsymbol{x},\boldsymbol{x}') = e^{\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{2\sigma^2}}$, $\sigma \in \mathbb{R}^+$.

In these simple kernels, one has to tune one or two hyper-parameters, which is usually done using cross-validation or validation error minimization with grid search. More sophisticated approaches are based on a number of estimates for cross-validation errors and gradient descent method in the kernel parameter space [12]. Another branch of research in this direction concerns learning the kernels from data, as it is introduced in [40].

2.4 Invariant Learning with Kernel Methods

It is often desirable to include some prior knowledge to the data model. By prior knowledge we mean any information about the task that is given in addition to the training samples.

In this general setting, the usual smoothness assumption (the smoothness of the modeling function in the vicinity of training samples) is also accepted based on a kind of prior knowledge. This assumption is actually one of the basic principles in Machine Learning [15]. However, more specific knowledge can also be used to improve the model quality. One of the most useful specific prior information is a knowledge on the invariances of the problem. We mean by the notion of invariance that the output of the learning system is unchanged while the inputs are transformed by some operator. Invariance under shifts and small rotations in the field of character recognition is a well-known example.

Among the growing activity in the application of kernel methods to real-world problems, the question of incorporating invariances into kernel-based models is of particular interest [24].

A general approach to the problem was considered in [8]. Suppose we deal with a sparse kernel model, where part of the α_i weights in (2.30) are equal to zero, and the samples $\boldsymbol{x}_q^{sv}$ with $\alpha_q \neq 0$ are the Support Vectors (SVs). Consider a model representation of the kind:

$$f(\boldsymbol{x}) = \sum_q \alpha_q K(\boldsymbol{x}_q^{sv}, \boldsymbol{x}) + b. \tag{2.34}$$

To provide the invariance of (2.34) to the small continuous transformation of the input vectors $\boldsymbol{x} \mapsto \boldsymbol{x} + d\boldsymbol{x}$, the following value should be unchanged:

$$d\rho = \sum_{q,i} \alpha_q d\boldsymbol{x}^i \partial_i K(\boldsymbol{x}_q^{sv}, \boldsymbol{x}), \tag{2.35}$$

where $\partial_i \equiv \partial/\partial \boldsymbol{x}_i$ and $\boldsymbol{x}^i$ is the i-th component of the vector $\boldsymbol{x}$. Assuming the following one-parameter transformation

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + \gamma \chi_i(\boldsymbol{x}), \;\; \gamma \in R^1, \tag{2.36}$$

defined with some function $\chi_i(\boldsymbol{x})$ the desired invariance will be satisfied if the kernel function $K(\boldsymbol{x}_q^{sv}, \boldsymbol{x})$ obeys the following equation:

$$\sum_i \chi_i(\boldsymbol{x}) \partial_i K(\boldsymbol{x}_q^{sv}, \boldsymbol{x}) \equiv 0. \tag{2.37}$$

For the case of multiple invariances one has to find the non-trivial integrals of the system of equations corresponding to (2.37).

Given all the independent integrals of the system of equations of the kind (2.37), one can construct the kernel, satisfying the desired invariances.

Although this approach is quite general, it suffers from a number of drawbacks. First, the decision function for the classification problem is the sign of (2.34). Thus, we are interested in keeping the sign of (2.34) unchanged under the transformation, and not the function (2.34) itself. Next, it is indirectly supposed that the set of Support Vectors $\{\boldsymbol{x}_q^{sv}\}$ and weights α_q are unchanged when the model is retrained on the data transformed by (2.36), which may not be the case even for small values of γ. Moreover, since one needs to find the integrals of the system which consists of equations such as (2.37), the proposed approach is not applicable practically for high-dimensional data, especially when dealing with multiple invariances.

Therefore, more specific methods have to be developed for practical use. The evident way consists in developing application-specific kernels for different types of data.In the presented research we, however, develop more general methods that can be applied in different areas. Our general (but not strictly obligatory) assertions to the desired methods are as follows. We would like the methods to be general enough to be used for most applications. The computational complexity of the algorithm (hence its speed) should not exceed the computational complexity of the base state-of-the-art methods (SVMs). Moreover, ideally we would like to exploit the existing optimization algorithms, developed for SVM training [47], [16].

The presented work includes modifications of state-of-the-art methods as well as the development of new approaches. More detailed description of the existing methods and the main directions of our research are presented in the next chapters.

Now we describe the state-of-the-art of invariant learning with kernels in more details.

2.4.1 Virtual Samples

Injecting virtual samples into the training database is a general state-of-the-art method for incorporating invariances that can be used in a number of ML algorithms [46]. Applied to SVMs, it can be modified into the Virtual Support Vector method (VSV) [60].

The idea of the method is to add virtual samples made only from the Support Vectors of the problem, transformed according to the desired invariance. The presence of the Virtual Support Vectors in the training set makes the solution more invariant to the considered transformation, while preventing too large training dataset (since SVs are often only a small part of the whole dataset).

Another related approach is known as Jittered Kernel and was proposed in [17]. The method combines generating artificial examples which are not included into the dataset but used for kernel function modification.

Regularization Theory.

The SRM principle has a strong analogy in regularization theory, developed in the 1960s by Tikhonov (see for example, [Tikhonov and Arsenin 1972]). Regularization is a method of solving problems by making some prior assumptions about the desired function. Instead of minimizing the loss function directly, one minimizes the so-called regularized functional containing two terms - the loss function and a special type of functional that reflects the chosen method of regularization.

Instead of minimizing the loss functional

$$R_{emp} = \frac{1}{\ell} \sum_{i=1}^{\ell} Q(y_i, f(\boldsymbol{x}_i, \alpha)), \tag{2.38}$$

which is analogous to the ERM principle, one instead minimizes the regularization functional

$$R_{reg} = \frac{1}{\ell} \sum_{i=1}^{\ell} Q(y_i, f(\boldsymbol{x}_i, \alpha)) + \lambda \Psi(\alpha), \tag{2.39}$$

where λ is an appropriately chosen constant, and Ψ is a *r*egularization term. Typically, Ψ is a measure of smoothness of $f(\boldsymbol{x})$, defined as some measure on the parameters α. If Ψ coincides with some measure of capacity, then the regularization method is analogous to the SRM principle. The constant λ controls the trade-off between smoothness of approximation (generalization ability or confidence interval) and accuracy of approximation (empirical risk). Moreover, the following links of regularization and invariant learning exist.

Regularization and Invariant Learning

The link between the approach of virtual samples and regularization was shown in [37]. For a learning algorithm based on the squared loss function it was shown that under a number of assumptions training the model with virtual examples in the training set is equivalent to adding a regularization term to the risk functional.

In this setting, it is considered that the original training data are randomly corrupted with some parametrized transformation. For example, for the case of local translational invariance, i.e. to the transformation of the input samples $\boldsymbol{x}_i \hookrightarrow \boldsymbol{x}_i + \delta\boldsymbol{x}_i$, where $\delta\boldsymbol{x}_i$ are considered to be normally distributed with variance σ_δ, it can be shown that the following regularization term has to be used:

$$\Psi(\boldsymbol{x}, \boldsymbol{w}) \sim \int {\sigma_\delta}^2 |\nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{w})|^2 p(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.40}$$

For the case of linear regression function $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$, it converges to the familiar term $\|\boldsymbol{w}\|^2$.

However, the assumptions needed for this equivalence are quite strong. For other loss functions (including the loss functions of Support Vector Machines for regression and classification) the existence of such kind of equivalence is a hard problem. Some links between SVMs and regularizartion framework are presented in [69], more links with emphasis on regression are revealed in [19].

The evident link between the VSV approach, jittered kernel and regularization is also a subject of our consideration. The general approach developed later in Chapter 3 is based on the objects "built" on samples and hence is a kind of limit case of the virtual sample approach.

2.4.2 Transformation Manifolds

Given an invariant transformation, we can consider that every sample is potentially equivalent to the manifold generated by this transformation. For this setting, a number of approaches were developed. Here we describe the two branches linked with the further research results presented in the subsequent chapters.

Tangent Distance

Tangent vectors were extensively used in the work of [67] to introduce invariances. A tangent vector is simply a linearization of the transformation at the given point. Then, a span

of all the tangent vectors is a linear tangent manifold. In the case of J local transformations one can linearly approximate the actual manifold $S(\boldsymbol{x}, \vartheta)$ as follows:

$$S_{linear}(\boldsymbol{x}, \vartheta) = \boldsymbol{x} + \sum_{j=1}^{J} (\vartheta_j - \vartheta_j^0) L_{\vartheta_j}(\boldsymbol{x}), \tag{2.41}$$

where $L_{\vartheta_j}(\boldsymbol{x})$ are local transformations of $\boldsymbol{x}$ defined by:

$$L_{\vartheta_j}(\boldsymbol{x}) = \left. \frac{\partial S(\boldsymbol{x}, \vartheta)}{\partial \vartheta_j} \right|_{\vartheta = \vartheta_0}. \tag{2.42}$$

Note that $L_{\vartheta_j}(.)$ are operators that generate the whole space of local transformations (a Lie algebra of local transformations). For example, three operators of X-translation, Y-translation and rotations about the origin produce a transformation (and a corresponding object) of all possible translations and rotations.

Tangent vectors $\ell_j(\boldsymbol{x})$ can be obtained by discretising the result of applying the operators L_{ϑ_j} to the sample $\boldsymbol{x}$. This is caused by real-life representation of images with picsels, while the transformation operators are defined on continious objects. The direct application of the Tangent Distance for SVM kernels was considered in [25]. This approach, however, has to be further developed in order to include non-local and non-linear transformations.

Invariance with Semi-Definite Programming

Considering the invariant manifold as a polynomial trajectory in the input space (which includes tangent manifold representation for single invariance), the algorithm for learning linear classifiers is based on semi-definite programming. The Semi-Definite Programming Machine (SDPM) [21] is able to find a maximum margin hyper-plane when the training examples are 2nd order polynomial trajectories instead of single points. The solution is found to be sparse in dual variables and allows to identify those points on the trajectory with minimal real-valued output as virtual support vectors. However, this optimization problem is time consuming to solve. Furthermore, the invariant transformations with more than one transformation parameter are hard to handle with this method.

2.4.3 Algorithms

In order to include the prior knowledge on invariances into the SVM classifier, one can modify the algorithm itself. As the margin is a key idea of the method, margin maximization can be constrained in the following way. Instead of maximing the margin by minimizing $\|\boldsymbol{w}\|^2$, consider the following:

$$(1-\gamma)\|\boldsymbol{w}\|^2 + \gamma \sum_{i=1}^{\ell} (\boldsymbol{w} \cdot \ell_j(\boldsymbol{x}))^2, \tag{2.43}$$

where $\ell_j(\boldsymbol{x})$ is a tangent vector (see previous section) and $\gamma \in [0, 1]$ controls the relative influence of the terms. The second term actually constrains the decision function of the linear SVM (2.11) to be smooth in the direction of tangent vector of the invariance, since:

$$f(\boldsymbol{x}_i + \ell_i(\boldsymbol{x})) - f(\boldsymbol{x}_i) = \boldsymbol{w} \cdot \ell_i(\boldsymbol{x}). \tag{2.44}$$

This is solved under the usual constraints (2.15) to enforce the SVM classifier to be invariant to local transformations. This can be generalized to the kernel version of SVM, while it requires complex computations (LU decomposition of the kernel matrix) and can not be handled on large training databases [11].

2.4.4 Invariant Features

An application-specific method of invariant learning consists in the use of invariant features. The relevant invariant features are first extracted from the raw data. Next, a "standard" kernel, such as RBF or polynomial, is used in the learning method. The prior knowledge on invariances is then included into the extracted features and not to the learning algorithm.

The areas of feature selection and kernel design are naturally linked. It is not possible to provide a boundary between these fields to discriminate where feature selection ends and kernel design starts. The feature selection process can be often included directly into the kernel calculation. In this work, we develop general methods devoted to invariance incorporation into kernel methods. Invariant features are not considered. However, we mention the most important current trends in the state-of-the-art in this field.

In the field of image analysis and object detection, SVM-based systems are used extensively. A number of recent developments in the image processing field are considered as potential candidates for baseline methods used for kernel design. The latter includes the approach of [65] based on the analysis of the objects using the statistics of parts, the approach of [38] where the Scale Invariant Feature Transform is introduced, that provides a way to produce a set of feature vectors invariant to basic image transforms, the approach of [42] based on detecting the interest points from the image and computing the set of features therein.

2.5 Conclusions

In this Chapter we presented the background for Machine Learning with Kernel Methods. The state-of-the-art of learning with prior knowledge and invariances was presented as well. In the next Chapters, the new approaches will be developed, which provide solutions for invariant learning with prior knowledge in different conditions. Given different representation of the invariances, the methods to incorporate this information will be presented. The presentation is organized in three main parts. First, in Chapter 3, we incorporate invariances by modifying the kernel functions for standard kernel algorithms. Next, in Chapter 4, we develop some approaches based on the construction of the new algorithms based on prior knowledge. Then, the kernel trick is applied. Finally, in Chapter 5, we adapt the approach from semi-supervised learning, which exploit unlabelled data to enforce the desired properties of the kernels.

Chapter 3

Invariant Kernels

This chapter presents a general method for incorporating prior knowledge on invariances into kernels. These kernels can be used in classification methods such as Support Vector Machines. It applies when the prior knowledge can be formalized by the description of an object around each sample of the training set, assuming that all points in the given object share the same desired class. A number of implementation techniques of this method, based on hard geometrical objects and soft objects based on distributions are considered. Tangent vectors are extensively used for object construction. Empirical results on one artificial dataset and two real datasets of Electro-Encephalogram signals and face images demonstrate the usefulness of the proposed method. The method can establish a foundation for an information retrieval and person identification systems. This chapter is mainly based on the publications [55], [50].

In this chapter we assume that the prior knowledge can be formalized as a mapping of a special kind. This mapping transforms each sample into an object in such a way that it includes the prior knowledge, similar to that done in the Tangent Distance approach of Simard et al. (1998) [67] applied to neural networks. The method does not lead either to enlarged training sets or to the modification of the cost function as opposed to other techniques. It simply exploits standard SVM optimization algorithms. It uses local models (objects) based on training samples. The restrictions for these models (i.e. the conditions for the object to be a tangent manifold, etc.) will be introduced as needed.

The rest of the chapter is organized as follows. The general idea is presented in Section 3.1, as well as two implementations for hard geometrical (Section 3.2.1) and soft distribution-based objects (Section 3.2.2). We give links to several other approaches in Section 3.3.1. Section 3.4 presents the experiments on artificial data where we illustrate the performance of the proposed method, and on two real datasets, where the first task is to classify EEG signals for a Brain-Computer Interface system and the second is devoted to a number of classification tasks in the area of face recognition and person identification. Section 3.5 completes this chapter with a discussion of the presented results.

3.1 From Samples to Objects

Suppose we have some understanding of our data that can be formalized as a transformation of the inputs that leaves the outputs unchanged. For example, in a 2D image classifi-

cation task we are often given the evident knowledge that small rotations and translations of the raw images do not affect the desired output class. Suppose the representation of the data (the set of features) allows us to describe the desired transformation as a mapping that leaves the outputs unchanged. The mapping applied to every sample produces a set of corresponding objects, which becomes a point of our consideration. In other words, we assume that given some understanding of the data we are able to generalize each sample into the equivalence class - the object in the input space. By doing this we aim to capture some prior similarities in the data. If this formalization is successfully performed, it is possible to deal with *objects* instead of samples when solving our particular learning problem. We will consider several kinds of such objects below.

3.1.1 Hard Objects

We define "hard" geometrical objects by the following transformation

$$\boldsymbol{x}_i \mapsto S_{\boldsymbol{x}_i} = \{\varphi_\vartheta(\boldsymbol{x}_i), \vartheta \in \Omega\}. \tag{3.1}$$

Function $\varphi_\vartheta(.)$ defines a set in the input space through the admissible set Ω of parameters ϑ. For example, one can consider segments or circles instead of points $\boldsymbol{x}_i$.

3.1.2 Soft Objects

Instead of using hard geometrical objects one can define an object as a local distribution of the kind

$$\boldsymbol{x}_i \mapsto p(\boldsymbol{x}|\boldsymbol{x}_i, \mathbf{r}_i), \tag{3.2}$$

where $\mathbf{r}_i$ is a vector of parameters of the local distribution $p(\boldsymbol{x}|\boldsymbol{x}_i, \mathbf{r}_i)$. This distribution is constructed in a way to describe the desired local transformations of a sample. It represents the probability that a given point $\boldsymbol{x}$ is in fact a (transformed) sample $\boldsymbol{x}_i$.

Though a uniform distribution on the bounded support can be considered as a hard object, we still discriminate the hard/soft cases due to the different underlying approaches used to define the kernel functions.

3.1.3 Objects Based on Tangent Vectors

One evident way to create objects from samples is to use the *tangent vector* approach. Tangent vectors were extensively used in the work of [67] to introduce invariances into neural networks. We will now present the notion of tangent vectors in a way suitable for further applications in image processing. A good intuition for the following equations lies in considering 2D images on the plane (ξ, ψ). The intensity of the image is defined by some function $U(\xi, \psi)$. It provides a high-dimensional input vector $\boldsymbol{x}$ for a given discrete set of coordinates (ξ, ψ).

Suppose the 2D transformation of the image plane t_ϑ we want to be invariant to is defined by the set of parameters ϑ in some region of $D \subset \mathbb{R}^2$:

$$t_\vartheta : D \subset \mathbb{R}^2 \mapsto t_\vartheta(D) \subset \mathbb{R}^2, \tag{3.3}$$

where ϑ is a J-dimensional vector which parametrises the transformation. This transformation is assumed to be differentiable with respect to ϑ and $(\xi, \psi) \in D$, and reduces to the

identity transformation for some value of ϑ^0. Then the object generated by this transformation and associated with an image U is defined by

$$S(U, \vartheta) = U \circ t_\vartheta^{-1},\ \vartheta \in \Omega, \tag{3.4}$$

where Ω is some admissible set of parameters ϑ. In the case of J local transformations one can linearly approximate $S(U, \vartheta)$ as follows:

$$S_1(U, \vartheta) = U + \sum_{j=1}^{J} (\vartheta_j - \vartheta_j^0) L_{\vartheta_j}(U), \tag{3.5}$$

where $L_{\vartheta_j}(U)$ are local transformations of U defined by:

$$L_{\vartheta_j}(U) = \left. \frac{\partial S(U, \vartheta)}{\partial \vartheta_j} \right|_{\vartheta = \vartheta_0}. \tag{3.6}$$

Note that L_{ϑ_j} are operators that generate the whole space of local transformations (a Lie algebra of local transformations). For example, three operators of X-translation, Y-translation and rotations about the origin produce a transformation (and a corresponding object) of all possible translations and rotations. *Tangent vectors* $\ell_j(\boldsymbol{x})$ can be obtained by discretising the result of applying the operators L_{ϑ_j} to the continuous image U which correspond to a discrete sample $\boldsymbol{x}$.

The examples of tangent vectors calculation for widely used transformations such as rotations and scaling are shown below:

- Rotation:
$$t_\vartheta = \begin{pmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{pmatrix}, \quad L_\vartheta^{rot} = \psi \frac{\partial}{\partial \xi} - \xi \frac{\partial}{\partial \psi}. \tag{3.7}$$

- Scaling:
$$t_\vartheta = \begin{pmatrix} 1+\vartheta & 0 \\ 0 & 1+\vartheta \end{pmatrix}, \quad L_\vartheta^{sc} = \xi \frac{\partial}{\partial \xi} + \psi \frac{\partial}{\partial \psi}. \tag{3.8}$$

If $\Omega = \mathbb{R}^J$, (3.4) is a J-dimensional differentiable manifold and (3.5) is a corresponding linear tangent manifold which is used in the tangent distance method of Simard et al. (1998) [67]. However, tangent vectors can hardly model the transformation of complex images, such as faces, providing acceptable results only for small values of ϑ_i. Therefore, the bounded set of parameters $\vartheta \in \Omega$ can be used. Since every parameter ϑ_i corresponds to a transformation we intend to be invariant to, then the choice of the set Ω defines the influence of this or that type of prior information on the invariances. Later, we will use tangent vectors for the construction of specific distributions which represent soft objects.

3.1.4 Objects Based on Sample Models

In general there is no need to build identical objects for every sample. Consider an example from Optical Character Recognition (OCR). There is no need to build an object which represents a set of symmetrically rotated characters for every sample of the database. A number of characters may appear to be rotated significantly already and the latter object representation would make them worse by rotating them unreasonably much.

So far, object construction can also be thought of as follows. Suppose we have a *model* $\Psi(.)$ for the samples we are dealing with. The model represents our knowledge of the sample's properties and can be of a very general kind. Note that it has to be output-independent to be used at the predictions stage. Given a sample from the dataset, the model transforms it into an *object* based on some estimated sample-dependent parameters: $object_i = \Psi(sample_i)$. Following the example from OCR, suppose we estimated the angle a digit character is rotated by, hence an object will be constructed to describe basically the rotation in the opposite direction.

However, practical implementation of this approach requires solving the tasks of application-dependent model construction and estimation of model parameters, which are rather complicated and will not be considered further here.

3.2 Tangent Vector Kernels

There are two major ways to make use of objects in kernel methods. Generally, one would like to formulate a criterion for a learning algorithm directly for objects. For example, a criterion can be to maximise the margin between objects of different classes. It will be considered later in Chapter 4.

We explore here another approach based on defining a kernel function for objects, which can be used in a standard algorithm like SVM.

3.2.1 Kernels for Hard Objects

Since we apply our knowledge directly and deal with objects in the input space, it is reasonable to deal with distance-based kernels that have clear intuitive interpretation as a measure of similarity.

Suppose one uses a distance-based kernel, for example the commonly used Gaussian Radial Basis Function (RBF) kernel:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{\frac{-\rho(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{2\sigma^2}}, \tag{3.9}$$

where sample-to-sample distance in the input space is defined by $\rho(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ and σ^2 is the variance of the kernel.

Substituting the object-to-object distances into the kernel, one includes the prior knowledge into the algorithm. The problem here is to provide a way to compute distances between objects efficiently. In the following subsections we give some simple examples of distances that can be derived analytically and calculated efficiently.

Linear Scaling

Consider the linear transformation

$$\boldsymbol{x} \mapsto G_{\boldsymbol{x}} = \{\vartheta \boldsymbol{x}, \vartheta \in [a, b]\} \tag{3.10}$$

where $a, b \in \mathbb{R}$. This transformation corresponds to a brightness change of the image (given a raw image representation) or to an amplitude scaling of the signal. Note that in

general for a finite range of ϑ this transformation cannot be taken into account by simple normalization of the input data.

Consider the distance given by:

$$\rho(\boldsymbol{x}, G(\hat{\boldsymbol{x}}))^2 = (\boldsymbol{x} - |\vartheta^*|_{[a,b]}\hat{\boldsymbol{x}})^2, \;\; \vartheta^* = \frac{\boldsymbol{x} \cdot \hat{\boldsymbol{x}}}{\hat{\boldsymbol{x}}^2}, \tag{3.11}$$

where $|g|_{[a,b]}$ is defined as a if $g < a$, b if $g > b$, and g otherwise. This is simply the distance between $\boldsymbol{x}$ and the segment $G_{\hat{\boldsymbol{x}}} = [a\hat{\boldsymbol{x}}, b\hat{\boldsymbol{x}}]$ of the line corresponding to the directing vector $\hat{\boldsymbol{x}}$. Though segment-to-segment distance can be easily derived, we will use a symmetrized sample-to-segment distance in the experiments below for an illustrative artificial example (Section 3.4.1) and the task of EEG signals classification (Section 3.4.2).

Translations

The second example is a particular case of translation invariance, i.e. the desired transformation is

$$\boldsymbol{x} \mapsto P_{\boldsymbol{x}} = \{\mathbf{e}_i t_i + \boldsymbol{x}; t_i \in [-t_i^{lim}, t_i^{lim}]\}, \tag{3.12}$$

where $\mathbf{e}_i$ are the basis vectors of the input space $\mathbb{R}^N$ and t_i^{lim} is the maximum allowed translation in dimension i, with $i = 1, 2, ..., N$. It corresponds to the mapping into an interior of the cuboid whose "center" is vector $\boldsymbol{x}$ and all the edges are parallel to the axes. This transformation corresponds to a speckle noise in the image and its use will be illustrated experimentally in Section 3.4.3.

The distance between vector $\boldsymbol{x}$ and cuboid $P_{\hat{\boldsymbol{x}}}$ is given by minimization of

$$\rho(\boldsymbol{x}, P_{\hat{\boldsymbol{x}}})^2 = \min_{\vec{t}} (\boldsymbol{x} - P_{\hat{\boldsymbol{x}}}(\vec{t}))^2, \tag{3.13}$$

over the set of parameters $\vec{t} = \{t_1, t_2, ...t_N\}$ and can be calculated as follows:

$$\rho(\boldsymbol{x}, P_{\hat{\boldsymbol{x}}})^2 = \sum_{i=1}^{N} ((\boldsymbol{x}^i - \hat{\boldsymbol{x}}^i) - |\boldsymbol{x}^i - \hat{\boldsymbol{x}}^i|_{[-t_i^{lim}, t_i^{lim}]})^2, \tag{3.14}$$

where $\boldsymbol{x}^i, \hat{\boldsymbol{x}}^i$ are the components of the corresponding vectors.

The distance between two objects defined by (3.12) can be similarly computed with

$$\rho(P_{\boldsymbol{x}}, P_{\hat{\boldsymbol{x}}})^2 = \sum_{i=1}^{N} \left| |\boldsymbol{x}^i - \hat{\boldsymbol{x}}^i| - 2t_i^{lim} \right|_{[0,\infty]}. \tag{3.15}$$

We will use this distance later for noisy image classification (Section 3.4.3).

Object to Object Distances

Using the sample-to-object distances, we take into account some prior knowledge but still use a kernel matrix that might not be positive definite. One can use an average of two sample-to-object distances to make the kernel matrix symmetric. Ideally, an object-to-object distance may be preferable, but its calculation is often quite a difficult task and

can not always be easily performed. For the considered examples it is possible to compute segment-to-segment and box-to-box distances (3.15).

In general, the computation of the Euclidean distance between the objects leads to a constrained optimization problem. Consider two objects $S(\boldsymbol{x}, \vartheta)$ and $\hat{S}(\hat{\boldsymbol{x}}, \hat{\vartheta})$. We approximate the Euclidean distance between them with the Euclidean distance between their linear approximations:

$$\rho(S, \hat{S})^2 \simeq \rho(S_1, \hat{S}_1)^2 = \min_{\vartheta, \hat{\vartheta}} \left(\boldsymbol{x} - \hat{\boldsymbol{x}} + \sum_{i=1}^{L} \vartheta_i \ell_{\vartheta_i}(\boldsymbol{x}) - \hat{\vartheta}_i \ell_{\hat{\vartheta}_i}(\hat{\boldsymbol{x}}) \right)^2, \tag{3.16}$$

subject to

$$\vartheta_i \in [\vartheta_i^{min}, \vartheta_i^{max}],\ \hat{\vartheta}_i \in [\hat{\vartheta}_i^{min}, \hat{\vartheta}_i^{max}]. \tag{3.17}$$

This problem can be considered as a Nearest Point Problem and a number of simple iterative methods such as Gilbert's or Mitchel-Demyanov-Malozyomov algorithms (which are also used for SVM training) can be applied for distance minimization (e.g. Keerthi et al. (2000), [34] and references therein). By controlling the maximum number of iterations one defines a trade-off between accuracy and speed. The unconstrained problem (3.16) corresponds to the Tangent Distance method. Its direct application for SVM kernels was considered by Haasdonk and Keysers, (2002), [25].

3.2.2 Kernels for Soft Objects

Given a soft object in a form of local distributions centered at each sample, one can apply a number of approaches developed in statistics for comparing two distributions. We first introduce here a special kind of distribution which makes use of tangent vectors. Next, we present some methods for defining the corresponding kernels.

Local Distributions based on Tangent Vectors

Suppose the transformation we want to be invariant to defines a differentiable manifold in the input space. Hence the tangent vectors can be defined as described above, and the whole set of tangent vectors can be used to model all the local linear transformations of the given image. Let us define the following function H which gives the measure of proximity of a given vector $\boldsymbol{x}$ to the linear span of some vector $\boldsymbol{x}'$ generated with a tangent vector ℓ_j:

$$H(\boldsymbol{x} \big| \boldsymbol{x}', \ell_j) = e^{-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2 \ell_j^2 - ((\boldsymbol{x}-\boldsymbol{x}')\cdot \ell_j)^2}{2\gamma_w^2 \ell_j^2}}, \tag{3.18}$$

where γ_w is the parameter related to the width of the proximity region.

The following distribution K_s describes a similarity between given sample $\boldsymbol{x}$ and an object based on sample $\boldsymbol{x}'$ generated by a set of corresponding tangent vectors $\{\ell_1, ...\ell_J\}$:

$$K_s(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{2\sigma^2}} \cdot \prod_{j=1}^{J} (\eta + H(\boldsymbol{x} \big| \boldsymbol{x}', \ell_j)) \tag{3.19}$$

where σ is a bandwidth and the real number $\eta \in [0, 1]$ defines the shape of the distribution.

Assuming $\eta = 0$ in (3.19) and applying normalization, one can reduce (3.19) to a standard Gaussian:

$$K_s(\boldsymbol{x},\boldsymbol{x}') = \frac{e^{-(\boldsymbol{x}-\boldsymbol{x}')^T L_{\boldsymbol{x}'}^{-1}(\boldsymbol{x}-\boldsymbol{x}')}}{(2\pi)^{N/2}|L_{\boldsymbol{x}'}|^{1/2}}, \text{ where :}$$
$$L_{\boldsymbol{x}'}^{-1} = \left(\frac{1}{2\sigma^2} + \frac{J}{2\gamma_w^2}\right) I - \sum_{j=1}^{J} \frac{\ell_j \ell_j^T}{2\gamma_w^2 \ell_j^2}, \tag{3.20}$$

where I is an identity matrix, and $|...|$ denotes the determinant. This representation will be extensively used below for kernel evaluation.

Tangent Vector Kernels

The simplest way to use the latter distribution for introducing invariances into kernel methods is to consider (3.19) as a one-sided (sample-to-object) similarity measure. Then, a two-sided kernel K_d can be obtained by taking the following average:

$$K_d(\boldsymbol{x},\boldsymbol{x}') = \frac{1}{2}(K_s(\boldsymbol{x},\boldsymbol{x}') + K_s(\boldsymbol{x}',\boldsymbol{x})). \tag{3.21}$$

The proposed kernel combines the advantages of both Virtual Support Vectors and Tangent Distance approaches. In this approach we not only analytically include the Virtual SV into the model (without putting them into the data), but also take into account all the linear combinations of invariant transformations of interest. Moreover, using all the tangent vectors which correspond to linear transformations, one can take into account all the possible local linear transformations of an image.

The proposed kernel (3.19)-(3.21) is not the only possible one to make use of the tangent vectors. Other kernels can be constructed in a similar way to the one presented by combining the terms (3.18) in a different manner.

3.3 Distribution-Based Tangent Vector Kernels

To be consistent in the sample-to-object approach, let us consider the distributions (3.18) defined for every sample. We introduce the Distribution-based Tangent Vector Kernel (DB TVK) as follows. The kernel can be obtained by measuring the overlap of two distributions that correspond to the object based on samples $\boldsymbol{x}$ and $\boldsymbol{x}'$. To do this we introduce the following kernel between two distributions:

$$K_B(\boldsymbol{x}',\boldsymbol{x}'') = \int K_s(\boldsymbol{x},\boldsymbol{x}')^\rho K_s(\boldsymbol{x},\boldsymbol{x}'')^\rho d\boldsymbol{x},\ 0 \le \rho \le 1, \tag{3.22}$$

which is a dot product in the space of functions $K_s(.,\boldsymbol{x}')$ and was called the probability product kernel in Kondor and Jebara (2004).

The closed form of $K_B(\boldsymbol{x}',\boldsymbol{x}'')$ can be obtained for a number of cases. For $\eta = 0$ and using equation (3.20), $K_B(\boldsymbol{x}',\boldsymbol{x}'')$ reduces to integration of Gaussians and can be expressed as follows:

$$K_B(\boldsymbol{x}',\boldsymbol{x}'') = (2\pi)^{\frac{(1-2\rho)N}{2}} \left|\hat{L}\right|^{\frac{1}{2}} |L_{\boldsymbol{x}'}|^{-\frac{\rho}{2}} |L_{\boldsymbol{x}''}|^{-\frac{\rho}{2}} \cdot \exp(-\tfrac{\rho}{2}\boldsymbol{x}'^T L_{\boldsymbol{x}'}^{-1}\boldsymbol{x}' - \tfrac{\rho}{2}\boldsymbol{x}''^T L_{\boldsymbol{x}''}^{-1}\boldsymbol{x}'' + \tfrac{1}{2}\hat{\boldsymbol{x}}^T \hat{L}\hat{\boldsymbol{x}}) \tag{3.23}$$

where $\hat{L} = (\rho L_{\boldsymbol{x}'}^{-1} + \rho L_{\boldsymbol{x}''}^{-1})^{-1}$ and $\hat{\boldsymbol{x}} = \rho L_{\boldsymbol{x}'}^{-1}\boldsymbol{x}' + \rho L_{\boldsymbol{x}''}^{-1}\boldsymbol{x}''$. A closed form equation for the distribution-based tangent vector kernel can also be derived for $\eta \neq 0$ and $\rho = 1$, which is more interesting but yields an even more cumbersome expression. We consider this case in the next section.

Making Distribution-based TVK practical

Direct implementation of the proposed DB TVK demands costly computations. Therefore, we propose here a practical way to compute (3.22). It consists of computing the approximation of the integral (3.22) for $\eta \neq 0$, $\rho = 1$ and $K_s(\boldsymbol{x}, \boldsymbol{x}')$ as presented in (3.19). Note that we fixed $\rho = 1$, hence the latter approximates the corresponding *expected likelihood* kernel. The following approximation can be used:

$$K_B(\boldsymbol{x}', \boldsymbol{x}'') = I_0 + (1 + \frac{\sigma^2}{2\gamma_w^2})^{\frac{1-D}{2}} \sum_{j=1}^{J} (I_j' + I_j'') + ..., \tag{3.24}$$

where

$$I_0 = e^{-\frac{(\boldsymbol{x}' - \boldsymbol{x}'')^2}{4\sigma^2}}. \tag{3.25}$$

The term I^0 correspond to the RBF kernel between samples $\boldsymbol{x}'$ and $\boldsymbol{x}''$. The terms I_j' and I_j'' correspond to the impact of j^{th} invariance of the samples $\boldsymbol{x}''$ and $\boldsymbol{x}'$ to the samples $\boldsymbol{x}'$ and $\boldsymbol{x}''$ correspondingly. The exact expressions of I_j' and I_j'' are quite cumbersome, which can be obtained as explained. The expansion of (3.22) with terms given by (3.19), $\rho = 1$ and $\eta = 1$ consist of the sum of products. One can neglect the terms which include the products of three and more exponents. Then the integration of the rest terms is analogous to (3.23). This approximation requires only $O(J \cdot N)$ operations to compute. The exact expression is out of the scope of our analysis, however. We present here the most important conclusion.

The impact of invariances reduces as the input dimension increases. Considering equation (3.24), one can see that with increasing dimension of the input space D second term in (3.24) decreases. For very high dimensional input spaces its influence vanishes, and the only term that matters is I_0. We've faced this problem in our experiments, which we describe later in Section 3.4.4.

3.3.1 Links with Kernel Jittering and Virtual SV

The distance between hard objects is a distance between some of the points the objects consist of. The points that give minimum to the distance effectively affect the model and can be considered as virtual samples. This allows interpretation of the described approach as a kind of virtual sample approach with automated choice of virtual samples, which may differ for every pair of objects. Furthermore, virtual samples can be used to replace tangent vectors with finite difference vectors. It appeared to be useful in our face classification experiments (see Section 3.4.3).

In analogy with the Virtual Support Vector approach of Scholkopf et al. (1996), [60],one can define objects based on pre-determined support vectors only to enhance the speed of the algorithm.

The method of kernel jittering was proposed by DeCoste and Burl (2000), [17]. It combines artificial sample generation and kernel function modification as follows. Consider

two samples, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and the corresponding non-jittered kernel function K_{ij}. Assume sample $\boldsymbol{x}_j$ could have been any of a set of values around $\boldsymbol{x}_j$ according to the desired transformation. Consider the transformed ("jittered") forms of the sample $\boldsymbol{x}_j$, including itself, and select one ($\boldsymbol{x}_{q^*}$) closest to $\boldsymbol{x}_i$ in the feature space according to the Euclidean distance in the feature space:

$$q^* = \arg\min_q \sqrt{K_{ii} - 2K_{iq} + K_{qq}}. \tag{3.26}$$

The new "jittered" kernel for the examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is simply K_{iq^*}. This idea can be interpreted as follows. Believing that transformed examples belong to the same class, kernel jittering corresponds to a kernel based on the distance between the sets generated from the examples by the allowed transformations.

The main drawback of the jittering approach is the need to do a lot of kernel calculations while selecting the minimal distance (3.26). The approach also requires that we do these calculations during the testing phase. Next, we have to note that the resulting value K_{iq^*} is not a kernel in its strict sense. The kernel matrix becomes non-symmetric and may appear to be non positive-definite. Thus, this approach may suffer from computational problems at the training phase.

The distance-based methods proposed above can be considered as an analytical jittering. It does not suffer from the drawbacks described above, however it introduces valuable restrictions on the allowed transformations.

3.4 Applications and Experiments

It is often difficult to formulate real-life problems in a way suitable for object definition in the input space. For example, it is difficult to define objects that correspond to the invariances of interest in image processing such as 3D rotations with changing lighting conditions. This is one of the drawbacks of the described approaches.

We present a series of experiments illustrating the proposed approaches. These are an artificial two-class classification task, a problem of EEG signals classification and a number of face image classification tasks.

3.4.1 Artificial Data

To illustrate the action of the considered methods, we used an artificial dataset generated to be invariant to (3.10). The goal is thus to illustrate the influence of the modified kernels on the decision boundary.

Figure 3.1 illustrates the training data for both classes and the decision boundaries obtained with the following algorithms: the left image shows the original SVM with RBF kernel ($\sigma = 0.2$); the center one shows an SVM with RBF kernel ($\sigma = 0.2$) and distance defined by (3.11) with $a = 0.5$, $b = 2$; finally, on the right we see an SVM with RBF kernel ($\sigma = 0.2$) and distance defined by (3.11) with $a = 0.01$, $b = 10$. The substantial difference between the presented solutions lies in the number of support vectors, which is 20 for the standard solution (left figure) and 8 for the modified one (right figure). Note, that given the knowledge of global scaling invariance (Equation (3.10) with $\vartheta \in \mathbb{R}$) one could obtain the right solution by simply using input normalization. However, this is not the case if the scaling is bounded (Equation (3.10) with $\vartheta \in [a, b]$).

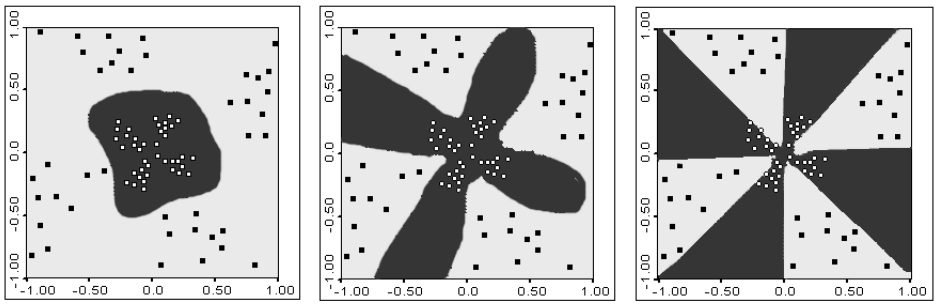


Figure 3.1. Artificial two-class classification problem. Black training points have to be discriminated against white training points. Left: Original decision function of an SVM with RBF kernel ($\sigma = 0.2$), Center: decision function using slightly modified kernel, Right: decision function facing full invariance.

3.4.2 EEG Signals Classification

The next series of experiments used EEG signals taken from the first competition devoted to Brain-Computer Interface system design. The competition was organized after the NIPS'01 Brain Computer Interface workshop. The task is to classify the signals that correspond to imaginary movements of the left or right hand. The original data consists of signals taken from different electrodes located on a human's head. The difference between two particular signals (from the C3 and C4 electrodes, according to the standard labeling) was taken as input for the algorithm. The data were resampled to 100Hz, the input dimension (the signal length) is 150. The dataset consists of 413 training and 100 testing samples. The details of data collecting and problem settings can be found at [http://newton.bme.columbia.edu/competition.htm].

Raw data usage may appear not to be the best way of carrying out classification. However, it was found to work well for SVMs. For example, the classification performance based on auto-regressive coefficients was significantly worse. The evident properties of these data are the invariances to the signal amplitude and the selection of the reference point of the "zero" level of the signal. These findings are also justified by the physical conditions of the EEG signal measuring process.

The results for the baseline SVM classifier based on Gaussian RBF kernel and SVMs with modified kernels (as derived in Sections 3.2.1 and 4.2.1) are presented in Table 3.1. The hyper-parameters of all the algorithms were tuned according to cross-validation on the training set. The obtained values are $C = 25$, $\sigma = 1500$. The invariance-defining parameters are $\gamma = 0.55$ for VRM-based kernel (this will be explained in Section 4.2.1), and for the kernel based on hard objects (segment-based SVM) the scaling range is $[0.5, 1.5]$.

Both methods provided an improvement of the classification performance according to the testing error. However, this improvement is hardly statistically significant (79% confidence only according to the standard t-test) since the size of the test set is only 100 samples. This is a basic disadvantage of the competition setting caused by difficulties in data collection.

Table 3.1. Experimental results on the EEG dataset.

	Algorithm	Testing Error, %
EEG	SVM	9
	Segment-based SVM	6
	VRM-based SVM	6

3.4.3 Face Recognition Experiments

The next example presented here deals with a real dataset obtained from a face detector. These are faces detected on every fifth frame of a movie using a face detector from Schneiderman and Kanade (2000). Image dimension is 81 by 81 and greyscale level is 8 bit. There were 2899 images in the database. The data is available at [http://www.robots.ox.ac.uk/ $\sim$ vgg/data]. We present an approach to the problem of binary classification of the main actor against all the other images captured. Hence, this task can be seen either as a person identification or an information retrieval task.

The training set consists of every tenth image of the database, while the testing set consists of all the other ones. We used the first thousand images of the database, ending up with training and testing sets of 100 and 900 samples correspondingly. Example images are presented in Figure 3.2.

Figure 3.2. Examples of original face images. Left: two random training samples. Right: three random testing samples. The labels for class membership are shown below the images.

Noisy Image Classification

To illustrate the use of the method described in Section 3.2.1 we have corrupted the images with an additive speckle noise. The noise is generated from a uniform distribution with zero mean and variance 30. Example images with noise are shown in Figure 3.3. Another noisy testing set was obtained by corrupting the clean testing set with the same noise, and "outliers": random 10% of the pixels were corrupted with uniform noise with zero mean and variance 70.

To show the performance of the method, we added noise to the testing set only. The objective is to obtain an algorithm robust to a known type of noise while given a clean training set only. Hence we are given a training set and a prior knowledge about the type of noise that occurs in the testing samples.

We used the raw image as input. Standard SVMs with Gaussian RBF kernel (3.9) were trained on the clean training set. The parameters were chosen according to the minimum

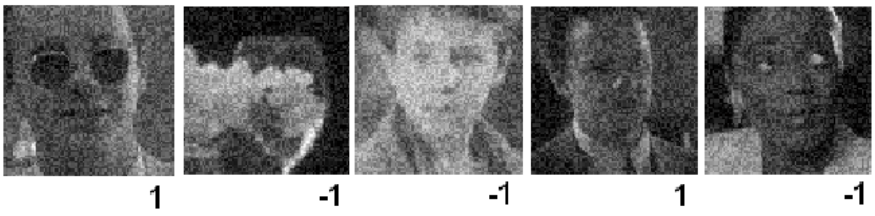


Figure 3.3. Examples of noisy face images. The labels for class membership are shown below the images.

of the cross-validation error. The parameters are: $\sigma = 3000$, $C = 100$. Classification error on the clean testing data is 9%, 17% on the noisy testing data and 37% on the noisy data with outliers. One possible solution to handle noise is to use denoising techniques to preprocess the testing data before applying the SVM classifier. Different denoising techniques such as Wiener filtering, median filtering and Gaussian bluring were used. The best result achieved was 14% of testing error for noisy data and 34% for the noisy data with outliers.

The SVM with an RBF kernel with the distance given by (3.15) was applied to the problem. The testing error for various values of the prior parameter t^{lim} is presented in Figure 3.4 for both noisy testing sets. The minimum of the testing error is achieved for the values of prior parameter t^{lim} which correspond to the standard deviation of the noise. The modified algorithm significantly (inside the 95 % confidence interval according to the t-test) outperforms standard SVM on the noisy testing data. However, testing error of the modified algorithm with $t^{lim} = 5$ gives 10.8% of the testing error on the clean testing data.

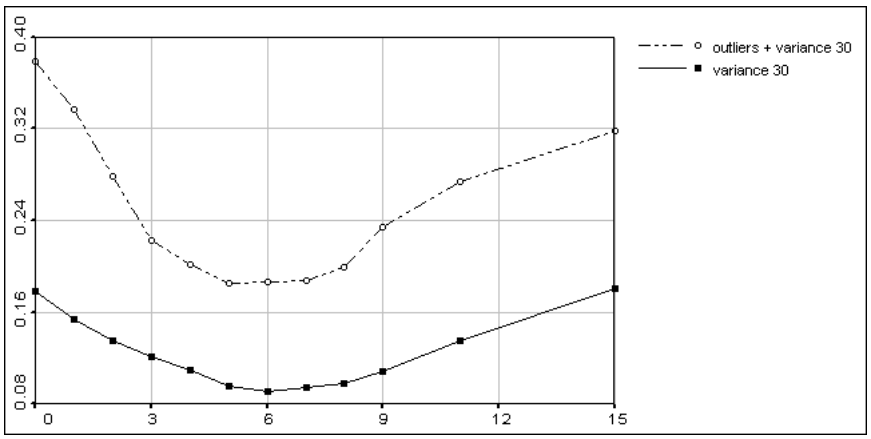


Figure 3.4. Testing error curve of the SVM with object-based kernel for both noisy testing sets. X-axis: t^{lim} parameter, Y-axis: testing classification error rate. Testing errors at $t^{lim} = 0$ (37% and 0.18%) correspond to standard SVM.

3.4.4 Invariant Face Images Classification

In order to test the proposed approaches of Section 3.2.2, we conducted experiments using images of the faces from the database described above. All the 2899 images of the database were used. We used subsets of 300 training and 2599 testing samples. The resolution was

decreased to 60x60 pixels.

We compared standard SVM with RBF kernel, Virtual Support Vector method, Kernel Jittering, and the proposed approaches of Sections (3.2.2) and (3.3). Two types of invariant transformations were studied: rotations (3.7) and scalings (3.8). Some considerations of practical implementation of the approaches are described below starting with tangent vectors evaluation.

Tangent Vectors and Finite Difference Vectors

There are some noticeable limitations in computing the tangent vectors. An input image has to be smooth enough to compute gradients that would approximate local transformations of the original image. The original method works well for binary images of digits, which were blurred with Gaussian filter for computing the gradients. We applied the method for our data using different Gaussian smoothing and found that the obtained approximation from these tangent vectors was not sufficient to describe real transformations. Instead we generated virtual samples by applying a finite desired transformation and used them for computing the finite differences that were used to approximate the tangent vectors. Example transformed images obtained by rotations with original gradient-based tangent vectors and finite differences are shown in Figure 3.5.

The first line in Figure 3.5 presents images obtained by applying direct calculation of tangent vectors according to (3.7). We can thus see that despite the accurate tuning of Gaussian filtering and other "tricks", only very local rotations are reasonable.

The second line in Figure 3.5 presents the original sample image $\boldsymbol{x}$ in the center; virtual samples obtained from $\boldsymbol{x}$ by applying rotations of 10 degrees are shown on the left and right of the figure. Let us denote them as $\boldsymbol{x} + \ell^{exp}_{left}$ and $\boldsymbol{x} + \ell^{exp}_{right}$. The intermediate images in between are $\boldsymbol{x} + 0.5\ell^{exp}_{left}$ and $\boldsymbol{x} + 0.5\ell^{exp}_{right}$.

The problem described here complicates the approach. On the other hand, the use of finite difference vectors allows for better modelling of the real-life invariances.

Figure 3.5. Two Types of Virtual Images. Top row: images obtained with tangent vector calculation. Bottom row: images obtained using finite differences.

Scaling and Rotational Invariances with TVK

Since this approach implied that left and right rotations correspond to different tangent vectors, we used the following modified Tangent Vector Kernel:

$$K_s^{fd}(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{2\sigma^2}} + \sum_{j=1}^{J} H(\boldsymbol{x}|\boldsymbol{x}', \ell_j) \cdot e^{-\frac{(\boldsymbol{x}-\boldsymbol{x}'-\ell_j)^2}{2\gamma_r^2}}, \tag{3.27}$$

where we introduced one extra parameter γ_r corresponding to the length of proximity region and replaced product with a sum. The experiments with modified kernels based on products (as presented in (3.19)) led to similar results. Note, that with this modification, one uses more than one "tangent vector" per invariance.

With a proper choice of parameters in (3.27) ($\gamma_w \sim \infty$, $\gamma_r = \sigma$), the resulted model is closely linked to VSV. The noticeable difference is that in the VSV approach every virtual sample is included in the decision function with its own weight, while in our case all the virtual samples form an object hence share the same weight.

The parameters of the algorithms were chosen according to the minimum of cross-validation error over the training set, resulting in $\sigma = 600$, $C = 100$. Parameters γ_w and γ_r in (3.27) can be chosen by the following heuristics: $\gamma_w \sim \sigma$, and $\gamma_r^2 \sim Var(\ell_{ij})$, i.e. the variance of tangent vectors. We used $\gamma_w = 500$ and $\gamma_r = 1000$.

Scaling and Rotational Invariances with DB-TVK

Despite of the curse of dimensionality problems described above, we used non modifyed Distribution-Based TVK, as it was introduced in Section 3.3. The parameters were as follows: $\sigma = 600$, $C = 100$, $\gamma_w = 1000$.

As it was mentioned above, DB-TVK has worse performance for high-dimensional input spaces. It is clearly seen from equations in Section 3.3, that the impact of "invariant" terms of the kernel reduces with dimensionality. However, we obtained reasonable results in the presented case study.

Experimental Results

Table 3.2 presents testing errors obtained with SVM with Gaussian RBF kernel (SVM), SVM trained with virtual samples (VSV SVM), SVM with jittered kernel (KJ SVM) and SVM with Tangent Vector and Distribution-based Tangent Vector Kernel (TVK SVM and DB-TVK SVM). We used the same virtual samples both for VSV and KJ SVM and for computing the finite difference vectors in TVK and DB-TVK. This is the reason of similar results obtained with all the methods. The improvement of the testing error obtained with developed TVK and DB-TVK kernels in comparison to the baseline SVM is statistically significant with a 95% confidence interval according to the standart t-test.

3.4.5 The Importance of Prior Knowledge for Small Datasets

Another interesting experiment is to show the relative importance of prior knowledge with respect to the amount of available training data. We thus split the data using every N-th sample of the entire data for training, while the rest of the data were used for testing.

Table 3.2. Face recognition testing error obtained with SVM with Gaussian RBF kernel, VSV SVM, SVM with jittered kernel and SVM with Tangent Vector and Distribution-based Tangent Vector Kernel.

	Algorithm	Testing Error, %	Training time, s
	SVM	11.2	5.2
	VSV SVM	9.8	24.0
	KJ SVM	10.0	25.2
	TVK SVM	9.7	12.5
	DB-TVK SVM	9.9	16.8

Figure 3.6 shows the testing errors obtained for these different partitions. The X-axis in Figure 3.6 corresponds to the logarithm of the training set size and the Y-axis corresponds to the testing error. As expected, when the number of training examples is very small, prior knowledge is of prime importance, while its importance eventually decreases with increased amount of training examples.

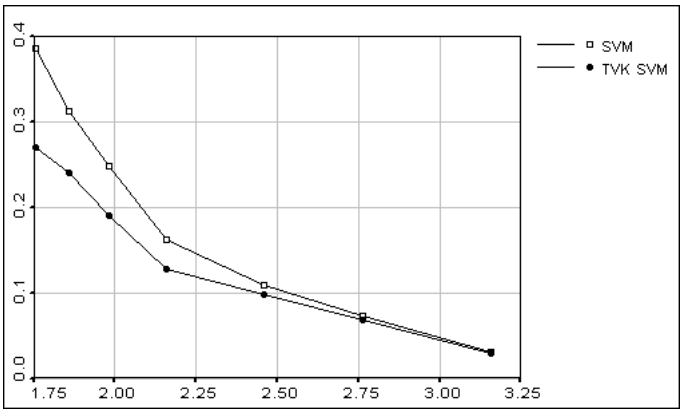


Figure 3.6. The performance of SVM with RBF and TVK kernels for different dataset sizes. The influence of prior knowledge increases with the decreasing training set size.

3.5 Discussion and Conclusions

The method of prior knowledge incorporation considered in this chapter consists in kernel modification and exploits the standard SVM algorithm. The main idea of the kernel construction is to consider an object in the input space: a set which can be derived for each training sample by applying all known invariant transformations. Then the kernel is defined for pairs of objects. Kernel calculation can appear to be a computationally expensive part of the algorithm, although in the considered examples it was not the case. The method does not lead to enlarging the training set.

The proposed approach has close links with the regularization framework. Loosely speaking, regularization is used to enforce smoothness of the function in the vicinity of the training points. For a learning algorithm based on the squared loss function it is shown by Leen (1995) in [37] that, under certain assumptions, the approaches of adding virtual sam-

ples to the training set and adding a regularization term to the cost function are equivalent. Our approach generalizes the virtual sample approach, and obviously it has regularizing properties.

Some similar approaches were recently proposed by Kondor and Jebara (2003), [35]. The idea there is to make a transition from samples to the sample-characterizing distributions which are then used for kernel definition. This approach mainly uses the distributions (objects) for data representation. As the evolution of the previous research, Kondor and Jebara (2004) [36] presented a similar sample-to-object framework. This transition step was used as an intermediate one to introduce probability product kernels. The aim of the presented research is to focus on the prior knowledge incorporation.

In conclusion, in this chapter we presented a general method to incorporate prior knowledge into kernel methods. It is based on modifying the setting of the problem by a transition from samples to objects, which are generated from them using some prior knowledge. We mainly considered these objects in the form of local distributions. Tangent Vectors were extensively used for the construction of the latters. Several methods of kernel definition were presented and tested in experiments on artificial and real-life data. This chapter is based on the results presented in the publications in [55], [50].

The next chapter is devoted to another general branch of including the prior knowledge into the learning algorithm. It will be done by modifying not the kernel but the algorithm according to the available prior knowledge.

Chapter 4

Invariant Kernel Algorithms

In this chapter, we present yet another approach to the problem of kernel learning with prior knowledge. Before, we modified the kernel function in accordance with the desired invariance. Despite the obtained improvement in the performance, the method was still subject to the curse of dimensionality. Now, we consider algorithms which include prior knowledge at the basic step. They will be constructed in a such a way that the kernel trick can be applied to produce the corresponding non-linear versions. This chapter is based on the publications [54], [53]. We start with the general considerations, devoted to the application of margin-based criteria to the objects in the input space.

4.1 Optimization with Constraints

Since we accept the sample-to-object approach, let us first consider the general setting of discriminating objects with a large margin. To maximize the margin between positively and negatively labeled objects (S_i^+ and S_i^- correspondingly) one has to solve the following optimization problem:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{4.1}$$

under the constraints:

$$\begin{array}{l} \min_{\boldsymbol{x}\in S_i^+} [\boldsymbol{w}\cdot\boldsymbol{x}+b] \geq 1,\ i\in I^+, \\ \max_{\boldsymbol{x}\in S_i^-} [\boldsymbol{w}\cdot\boldsymbol{x}+b] \leq -1,\ i\in I^-, \end{array} \tag{4.2}$$

where we simply denoted the index sets of positive and negative examples by I^+ and I^-. These constraints actually mean that we would like that the whole object to be outside the respective side of the margin. To solve this problem let us first replace the (4.2) with the only unifying constraint

$$\min_{\boldsymbol{x}\in S_i} y_i\,[\boldsymbol{w}\cdot\boldsymbol{x}+y_i b] \geq 1,\ \forall i. \tag{4.3}$$

So far, in primal form this is a constrained quadratic optimization. The type of constraints is defined by the type of the object considered. By selecting an appropriate object, this optimization can be reduced to some known (and reasonable to solve) problem. For example, the following sets were considered. Graepel et al. [21] considered quadratic tangent trajectories instead of samples. It actually leads to Semi-Definite Programming optimization. Using polygones as in [18] leads to a more complicated Quadratic Programming problem.

Considering ellipses (or Gaussian distributions, as we shall see later) leads to the Second Order Cone Programming problem [3], [4]. These optimization problems are difficult (time and memory consuming) to solve, and an iterative procedure to obtain an appropriate approximate solution is often desirable.

Let us consider the general setting, which illustrates the use of an iterative procedure for solving some kinds of margin maximization problems for objects. Introducing the non-negative multipliers $\alpha_i,\ i \in I$ for the constraints (4.3) we form the Lagrangian:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, b, \alpha) &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i\in I}\alpha_i\left[\min_{\boldsymbol{x}\in S_i} y_i\boldsymbol{w}\cdot\boldsymbol{x} + y_i b - 1\right] \\ &= \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i\in I}\alpha_i - b\sum_{i\in I} y_i\alpha_i - \sum_{i\in I} y_i \min_{x\in S_i}\left[y_i\boldsymbol{w}\cdot\boldsymbol{x}\right]\end{aligned} \tag{4.4}$$

which has to be maximized with respect to $\boldsymbol{w}$ and b and minimized with respect to α_i. The last term of (4.4) defines the influence of the prior knowledge (incorporated with objects) on the algorithm. Particularly, given the object S_i which consists of only the data sample $\boldsymbol{x}_i$ implies

$$\min_{\boldsymbol{x}\in S_i}\left[y_i\boldsymbol{w}\cdot\boldsymbol{x}\right] = y_i\boldsymbol{w}\cdot\boldsymbol{x}_i, \tag{4.5}$$

and we arrive at standard large margin classifier.

The problem to solve at the moment is to compute the last term of (4.4). Let us consider the following object, which corresponds to the sample $\boldsymbol{x}_i$:

$$S_i = \boldsymbol{x}_i \cup \{\boldsymbol{x}_i + \ell_i^j,\ j = 1, \ldots, J\}, \tag{4.6}$$

where ℓ_i^j is the finite difference (tangent) vector of the j^{th} desired transformation. This is actually the substitution of every sample with a cloud of virtual samples. With this, we obtain the following optimization problem:

$$\min_{\boldsymbol{x}\in S_i}\left[y_i\boldsymbol{w}\cdot\boldsymbol{x}\right] = y_i\boldsymbol{w}\cdot\boldsymbol{x}_i + \min\left[y_i\boldsymbol{w}\cdot\sum_j \ell_i^j\right] = y_i\,\boldsymbol{w}\cdot\boldsymbol{x}_i + \left(-\sum_j |\boldsymbol{w}\cdot\ell_l^j|\right). \tag{4.7}$$

Back substitution to (4.4) and differentation in b and $\boldsymbol{w}$ (according to the Kuhn-Tuker conditions) gives

$$\sum_{i\in I} y_i\alpha_i = 0, \tag{4.8}$$

$$\boldsymbol{w} - \sum_{i\in I} y_i\alpha_i\boldsymbol{x}_i - \left(-\sum_{i\in I}\alpha_i\sum_j \ell_i^j \operatorname{sign}\left[\boldsymbol{w}\cdot\ell_i^j\right]\right) = 0. \tag{4.9}$$

The first condition is equivalent to the analogous one in Large Margin Classifier. The second condition has to be used to obtain the dual form for $\boldsymbol{w}$. To combat this problem, we organize an iterative procedure for computing $\boldsymbol{w}$, starting from some initial value $\boldsymbol{w}^0$. The first iteration gives:

$$\boldsymbol{w} = \sum_{i\in I} y_i\alpha_i(\boldsymbol{x}_i + \delta\boldsymbol{x}_i) \tag{4.10}$$

where we denoted

$$\delta \boldsymbol{x}_i = -y_i \sum_j \ell_i^j \text{sign}\left[\boldsymbol{w}^0 \cdot \ell_i^j\right]. \tag{4.11}$$

So far, we are able to obtain the expression for decision function

$$f(\boldsymbol{x}) = \sum_{i \in I} y_i \hat{\alpha_i} \boldsymbol{x} \cdot (\boldsymbol{x}_i + \delta \boldsymbol{x}_i) + b, \tag{4.12}$$

where $\hat{\alpha_i}$ gives maximum to the dual form of the optimization problem:

$$\hat{\alpha_i} = \arg\max_{\alpha} \sum_{i \in I} \alpha_i - \frac{1}{2} \sum_{\substack{i \in I \\ j \in I}} y_i y_j \alpha_i \alpha_j (\boldsymbol{x}_i + \delta \boldsymbol{x}_i) \cdot (\boldsymbol{x}_j + \delta \boldsymbol{x}_j) \tag{4.13}$$

under the following constraints:

$$\begin{array}{l} \sum_{i \in I} y_i \alpha_i = 0, \\ \alpha_i \geq 0, \quad i \in I. \end{array} \tag{4.14}$$

We first note the meaning of the additive term (4.11): it shifts the "effective" sample to the closest position to the separating hyperplane, as it was originally implied by Eq. (4.3). Let's note, that this fact can be used to develop a strategy for generating the "optimal" virtual samples, i.e. a small number of samples which shift the decision surface. This can be done, for example, by selecting the virtual samples which violates the constraints of the primal optimization problem most.

We will show in Section 4.3 that this property also applies when a sample is replaced by a distribution. This will be used to introduce an iterative procedure for solving the optimization problem. We start, however, with an obvious way to deal with distributions as input objects for the classifier, which is based on Vicinal Risk Minimization principle.

4.2 Vicinal Risk Minimization

Vapnik (2000) considered local distributions (soft objects) instead of samples to introduce the Vicinal Risk Minimization (VRM) learning principle (see Section 2.1.2). The following Vicinal Support Vector algorithm is obtained by Vapnik (2000):

$$f(\boldsymbol{x}) = \sum_{i=1}^{\ell} \beta_i^* D(\boldsymbol{x}, \boldsymbol{x}_i) + b, \tag{4.15}$$

where the β_i^* coefficients are such that

$$\beta^* = \arg\max_{\beta} \sum_{i=1}^{\ell} \beta_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \beta_i \beta_j M(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{4.16}$$

subject to the constraints:

$$\begin{array}{l} \sum\limits_{i=1}^{\ell} y_i \beta_i = 0, \\ 0 \leq \beta_i \leq C. \end{array} \tag{4.17}$$

Functions $D(\boldsymbol{x}, \boldsymbol{x}_i)$ and $M(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are one- and two-vicinal kernels correspondingly:

$$D(\boldsymbol{x}, \boldsymbol{x}_i) = \int K(\boldsymbol{x}, \boldsymbol{x}') p(\boldsymbol{x}'|\boldsymbol{x}_i, r_i) d\boldsymbol{x}', \tag{4.18}$$

$$M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \iint K(\boldsymbol{x}, \boldsymbol{x}') p(\boldsymbol{x}|\boldsymbol{x}_i, r_i) p(\boldsymbol{x}'|\boldsymbol{x}_j, r_j) d\boldsymbol{x} d\boldsymbol{x}'. \tag{4.19}$$

Now we show how to use specific density functions $p(\boldsymbol{x}|\boldsymbol{x}_i, r_i)$ defined on the non-symmetrical support that describes the invariance to the desired transformation. By this we enforce the VRM-based learning algorithm to obey the invariance's properties.

4.2.1 Scaling Invariance

Let us now present a simple example. To obtain the scaling invariance of the training samples $\boldsymbol{x}_i$, consider the following vicinity density function:

$$p(\boldsymbol{x}|\boldsymbol{x}_i, \gamma) = \frac{1}{\sqrt{2\pi}\gamma} \int \delta(\boldsymbol{x} - (1-\alpha)\boldsymbol{x}_i) e^{-\frac{\alpha^2}{2\gamma^2}} d\alpha, \tag{4.20}$$

where δ is the delta function, and the γ parameter defines the width of the vicinity and, hence, the influence of scaling invariance.

Substituting (4.20) in both (4.18) and (4.19) using the standard isotropic RBF kernel function given in (3.9) with the bandwidth parameter σ gives:

$$D(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{\sigma}{\kappa} e^{-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^2}{2\kappa^2}} e^{-\frac{\gamma^2}{2\sigma^2\kappa^2}(\boldsymbol{x}^2\boldsymbol{x}_i^2 - (\boldsymbol{x},\boldsymbol{x}_i)^2)} \tag{4.21}$$

and

$$M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sigma^2}{\eta} e^{-\frac{\sigma^2(\boldsymbol{x}_i-\boldsymbol{x}_j)^2}{2\eta^2}} e^{-\frac{\gamma^2}{\eta^2}(\boldsymbol{x}_i^2\boldsymbol{x}_j^2 - (\boldsymbol{x}_i,\boldsymbol{x}_j)^2)}, \tag{4.22}$$

where the following definitions were used:

$$\kappa^2 = \gamma^2 \boldsymbol{x}_i^2 + \sigma^2, \tag{4.23}$$

$$\eta^2 = \gamma^2\sigma^2(\boldsymbol{x}_i^2 + \boldsymbol{x}_j^2) + \gamma^4(\boldsymbol{x}_i^2\boldsymbol{x}_j^2 - (\boldsymbol{x}_i, \boldsymbol{x}_j)^2) + \sigma^4. \tag{4.24}$$

The resulting kernels are still RBF-based. The "effective" kernel bandwidth depends both on the σ and γ parameters and on the samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Experiments with VRM

Experimental results obtained when using VRM-based kernels were mentioned before in Section 3.4, Table 3.1. They include artificial data classification and EEG signals classification. The performance of the method appeared to be similar to the one which uses Tangent Vector kernels described in the previous chapter. The reason for that is as follows. One can note the similarity of (4.21)-(4.22) to the kernel (3.20) presented before. Thus, with a correct choice of parameters, the results of the methods in the case of scaling invariance coincide.

Generally, VRM-based kernel classifier considers the distributions by taking averages of the kernels, as it is clealy seen in Eqs. (4.18), (4.19). We would like to obtain a method which is more precise in terms of classification of distributions by their domain. The next section is devoted to the latter.

4.3 A Kernel Classifier for Distributions

In this section we present a new algorithm for classifying distributions. The algorithm combines the principle of margin maximization and a kernel trick, applied to distributions. Thus, it combines the discriminative power of support vector machines and the well-developed framework of generative models. It can be applied to a number of real-life tasks which include data represented as distributions. The algorithm will be applied for introducing some prior knowledge on invariances into a discriminative model. We illustrate this approach in details for the case of Gaussian distributions, using a toy problem. We also present experiments devoted to the real-life problem of invariant image classification. We start with the background and motivation for our developments.

Large margin classifiers such as Support Vector Machines (SVMs) have shown to yield state-of-the-art performance in various classification tasks [7]. Moreover, classification tasks are known to be in general better solved by discriminant approaches (which aim at providing a model that minimizes some error on the training set of positive and negative examples), than by generative approaches (which aim at providing class-specific models that maximize the likelihood of the corresponding class training data).

On the other hand there are still several well-known classification tasks for which the current state-of-the-art is based on generative models. This is in general due to one of many possible reasons. For instance, in speech recognition, for which the best solutions are based on Hidden Markov Models and Gaussian Mixture Models trained by EM [57], one faces datasets of potentially millions of frame examples, which cannot be handled by SVMs given the (at least) quadratic training time and space complexity.

Another example is the task of speaker verification, which consists of determining whether the voice of a given person corresponds to the claimed ID. One problem with this task is that one can in general only collect very small (and thus non representative) amount of data of a given client, but very large datasets of potential impostors, which makes the problem highly imbalanced. State-of-the-art systems are thus based on the likelihood ratio of generative models of the two classes, where the client model is often adapted from the impostor model, given the lack of client specific data [39].

Based on these facts, there have been several attempts at integrating generative and discriminant approaches into one framework. One such attempt is based on the Fisher Kernel [29], where an SVM with a specific kernel is trained on examples which are the derivative of the log likelihood of the generative models of each class with respect to the parameters of the models.

Furthermore, in many classification tasks, feature extraction procedures sometimes result in huge sets of features, which can be hardly processed in the raw representation. One solution often used is to model them with distributions. In the field of image processing, this is the case for invariant features, extracted at some points of interest of an image. This approach is extensively used in object categorization problems [38]. One of the direct solutions for this problem is to build a kernel classifier (SVM) by defining the kernel over distributions, using either KL-divergence or similar methods [43].

Facing these problems, we feel it is worth developing a discriminative classifier which would directly deal with generative models, i.e. distributions. The approach presented in this paper exploits some nice performance of margin-based methods by constructing a maximum margin solution for a set of distributions labeled into two classes.

The main field of applications in the scope of the presented research concerns invariant learning. As it was introduced before, it is reasonable to deal with the whole set of patterns which can be obtained from every given training sample by applying some transformation (such as translation or rotation). A "soft" representation of these manifolds as distributions will be used to apply the developed method for handling invariances.

The rest of the chapter is organized as follows. In Section 4.3.1, the notion of margin maximization for distributions is defined. Some general facts, which provide a foundation for an approximate approach, described in Section 4.3.2, are also presented. Section 4.3.3 justifies the proposed approximations. Next, we consider the particular important case of Gaussian distributions in Section 4.3.4. Section 4.3.5 is devoted to the invariant image classification experiments, followed by discussion and conclusions to the chapter in Section 4.3.6.

4.3.1 Margin Maximization for Distributions

The margin maximization principle is based on results from the Statistical Learning Theory [74], and provides a way to minimize the complexity of the model by bounding the VC-dimension of the modeling function (see Chapter 2 for introduction to SLT). Intuitively, the same approach can be used for other learning tasks. We thus now present a definition of margin for distributions, and provide a way for constructing learning algorithms. The proof of the margin maximization principle for the considered problem is out of the scope of this research.

Suppose one is given a training set of L probability distribution functions $p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i)$, centered at $\boldsymbol{x}_i$ and specified by some parameters $\boldsymbol{r}_i$. We also associate some label y_i for each distribution. These are {+1, -1} for binary classification problem.

Linear Decision Functions

Consider the set of linear decision functions $\{f = \boldsymbol{w}\boldsymbol{x} + b\}$, where $\boldsymbol{w}$ is a weight vector, and b is a constant threshold. The actual decision is usually taken according to $\mathit{sign}(f)$.

Consider the optimization problem:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{L}\xi_i \tag{4.25}$$

subject to the following constraints:

$$\int\limits_{y_i(\boldsymbol{w}\boldsymbol{x}+b)\geq 1} p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i)d\boldsymbol{x} \geq \eta - \xi_i, i = 1,...L, \tag{4.26}$$

$$\xi_i \geq 0, i = 1,...L. \tag{4.27}$$

The first constraint corresponds to the fact that η-quantile of the distribution lies outside the margin, not taking into account the slack variable ξ_i. These slack variables are equivalent to the analogue trick done in soft margin formulation of the Support Vector Machine, Eqs. 2.24, 2.25.

4.3.2 Optimization Problem

The method of Lagrange multipliers can be applied to solve the problem stated in Section 4.3.1. Introducing the Lagrange multipliers $\{\alpha_i\}$ and $\{\beta_i\}$, one obtains the optimization problem of finding the saddle point of Lagrangian. Differentiation in b and $\boldsymbol{w}$ gives:

$$\sum_{i=1}^{L} \alpha_i \frac{\partial}{\partial b} \int\limits_{y_i(\boldsymbol{w}\boldsymbol{x}+b)\geq 1} p(\boldsymbol{x}|\boldsymbol{x}_i, \boldsymbol{r}_i) d\boldsymbol{x} = 0, \tag{4.28}$$

$$\sum_{i=1}^{L} \alpha_i \frac{\partial}{\partial \boldsymbol{w}} \int\limits_{y_i(\boldsymbol{w}\boldsymbol{x}+b)\geq 1} p(\boldsymbol{x}|\boldsymbol{x}_i, \boldsymbol{r}_i) d\boldsymbol{x} = \boldsymbol{w}. \tag{4.29}$$

Analogously to the standard SVM, the multipliers β_i vanish, resulting in a box-type constraints for the weights α_i: $0 \leq \alpha_i \leq C$. Next, introducing the following notation:

$$I(\boldsymbol{w}, \boldsymbol{x}_i, t) = \int\limits_{y_i(\boldsymbol{w}\boldsymbol{x}+b)=1-t} p(\boldsymbol{x}|\boldsymbol{x}_i, \boldsymbol{r}_i) d\boldsymbol{x}, \tag{4.30}$$

one yields the following optimization problem:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{L} \int\limits_{\xi_i}^{\infty} I(\boldsymbol{w}, \boldsymbol{x}_i, t) dt, \tag{4.31}$$

$$s.t. \sum_{i=1}^{L} \alpha_i y_i I(\boldsymbol{w}, \boldsymbol{x}_i, \xi_i) = 0, \tag{4.32}$$

$$\sum_{i=1}^{L} \alpha_i y_i \boldsymbol{x}_i^* I(\boldsymbol{w}, \boldsymbol{x}_i, \xi_i) = \boldsymbol{w}, \tag{4.33}$$

$$0 \leq \alpha_i \leq C. \tag{4.34}$$

Generally, this problem can not be reduced to the dual variable formulation since there is no closed form solution for $\boldsymbol{w}$. However, for a number of applications and particular types of distributions we will approach this problem by an approximate solution. Let us note, however, that for the case of traditional data samples, $p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i)$=$\delta(\boldsymbol{x}$-$\boldsymbol{x}_i)$, the problem (4.31)-(4.34) reduces to the standard soft margin SVM.

Iterative Solution

The general approach for solving the optimization problem (4.25)-(4.27), or its equivalent (4.31)-(4.34), is to apply an iterative procedure to obtain an approximate solution. This type of optimization approach has been described in [2], and applied in [4]. Generally, the nature of SVM-related methods is that they try to find the Support Vectors, i.e. the samples which lie closest to the discrimination surface. When discriminating some subsets $S(\boldsymbol{x}_i)$, constraints of the type $\max\limits_{\boldsymbol{x}\in S(\boldsymbol{x}_i)} [y_i(\boldsymbol{w}\boldsymbol{x}+b)] \geq 1-\xi_i$ can be used. See [21, 18, 3] for

examples of such solution for different types of $S(\boldsymbol{x}_i)$. Solving problems with this type of constraints is roughly equivalent to the task of finding the "optimal" or "effective" sample from the subset.

Here we show that a similar approach holds for the case of distributions. Let us consider the following result.

Lemma. Consider the optimization problem (4.25)-(4.27). There exists a set of samples $\{\boldsymbol{x}_i^*,\ i=1,\ldots L\}$ such that the optimal separating hyper-plane $\boldsymbol{w}^*$ for the set $\{\boldsymbol{x}_i^*\}$ coincides with the solution of the problem (4.25)-(4.27).

Proof. If the dimensionality of the feature space is less than the number of (non degenarative) samples, then the proof is trivial. Otherwise, for high (infinite) dimensional feature spaces, let $\boldsymbol{w}$ be the solution of (4.25)-(4.27). Since $p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i)$ is a p.d.f., then for any $\boldsymbol{w}$, and any fixed $\boldsymbol{x}_i$ and $p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i)$, function $I(\boldsymbol{w},\boldsymbol{x}_i,t)$ is continuous and monotonically increases with t. Then, according to the Weicherstrass theorem, there exists $\boldsymbol{x}_i^*$ such that $I(\boldsymbol{w},\boldsymbol{x}_i,-y_i\boldsymbol{w}\boldsymbol{x}_i^*)=\eta$. Thus, the optimization problem (4.25) can be reformulated in terms of minimizing the regularized risk functional with the cost function $c(.,.,.)$ that only depends on the choice of $\boldsymbol{x}_i^*$ for any fixed $\boldsymbol{x}_i$, and $p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i)$: $c(\boldsymbol{x}_i,y_i,\boldsymbol{w}) = C\sum_{i=1}^{L}\max(0,\eta - I(\boldsymbol{w},\boldsymbol{x}_i,-y_i\boldsymbol{w}\boldsymbol{x}_i^*))$. Thus, there exists a representation $\boldsymbol{w} = \sum_{i=1}^{L}\alpha_i\boldsymbol{x}_i^* + b$ which coincides with the solution of the problem (4.25)-(4.27).

The considered result, however, does not provide a way for finding neither $\boldsymbol{x}_i^*$ nor α_i in the desired representation.

A general iterative scheme includes iterations through a series of (currently) optimal samples for a given approximation to the hyper-plane. Then the margin is maximised again for the modified samples, etc. The major disadvantage of this type of approaches is the convergence of the described procedure. Even if it is possible to prove that these iterations converge, the rate of convergence is unknown and may appear to be unreasonably low. One of the successful attempts is [72], where the rate of convergence in a similar problem was estimated. However, the problem considered in [72] is a quite specific case, where iterations are carried out in the context of structured *outputs*.

We thus propose a simple 2-step method to obtain an approximate solution to (4.25)-(4.27). While we do not provide the proof of convergence, it is justified by the empirical evidence in our experiments.

4.3.3 Hyper-plane Projection Method

From now on, let us deal with the kernelized version of the proposed algorithm. Let $K(.,.)$ be a positive definite reproducing kernel. Let some $(\boldsymbol{w}_0, b_0)$ define the optimal separating hyper-plane (in the feature space) for the training set of means $\{\boldsymbol{x}_i, y_i\}$ in the feature space induced by $K(.,.)$. Actually, this is given by the set of non-zero Lagrange multipliers $\{\alpha_i\}$, obtained by solving the standard SVM optimization. The proposed scheme is as follows:

- solve a standard SVM optimization problem for the means $\boldsymbol{x}_i$. The solution is $(\boldsymbol{w}_0, b_0)$;
- calculate the projections of $p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i)$ on $\boldsymbol{w}_0$. This results in a 1-D optimization problem (see details below);

- solve the 1-D problem according to the given value of η;
- compute the inverse projection. This results in a modified training set $\boldsymbol{x}_i^*$ (see details below);
- solve a standard SVM optimization problem for the samples $\boldsymbol{x}_i^*$.

The detailed explanation of the projection steps is presented below. Please also refer to Figure 4.1 for an illustration.

Direct Projection

Consider the following averages in the feature space, which provide the means and variances of some 1-D distribution $\pi(\chi|\mu_j, \sigma_j)$.

$$\mu_j = E[\boldsymbol{w}_0\Phi(\boldsymbol{x}_j) + b_0] = \sum_{i=1}^{L} y_i\alpha_i \int_{\mathrm{X}} K(\boldsymbol{x}_j, \boldsymbol{x}_i)p(\boldsymbol{x}|\boldsymbol{x}_i, \boldsymbol{r}_i)d\boldsymbol{x} + b_0, \tag{4.35}$$

$$\sigma_j^2 = E[(\boldsymbol{w}_0\Phi(\boldsymbol{x}_j) - \mu_j)^2] = \sum_{i,k=1}^{L} y_i y_k \alpha_i \alpha_k \int_{\mathrm{X}^2} K(\boldsymbol{x}_i, \boldsymbol{x}_k)p(\boldsymbol{x}|\boldsymbol{x}_i, \boldsymbol{r}_i)p(\boldsymbol{x}'|\boldsymbol{x}_k, \boldsymbol{r}_k)d\boldsymbol{x}d\boldsymbol{x}' - \mu_j^2. \tag{4.36}$$

These 1-D p.d.f.s correspond to $p(\boldsymbol{x}|\boldsymbol{x}_j,\boldsymbol{r}_j)$ being projected to the 1-D subspace defined by $\boldsymbol{w}_0$. Given these projections, the constraints (4.26) can be (currently) satisfied by taking the χ_j in 1-D space such that

$$\int_{y_j f(\boldsymbol{x}) \geq \chi_j} \pi(\chi|\mu_j, \sigma_j)d\chi \geq \eta. \tag{4.37}$$

It can be solved easily and results in some threshold constant c_η^j such that $\chi_j = f(\boldsymbol{x}_j^*) = c_\eta^j$. The difficulty arises for the original samples that have been classified incorrectly. Currently, we propose to neglect these samples. The reasoning is simple: one would not like to update the classification based on doubtful sample distributions, which means have not been classified correctly.

Inverse Projection

Next, given the set of χ_j one has to find an inverse projection of χ_j into the feature space. Obviously, this transformation is not unique and some criteria are required to define it. At this step, it is hard to control the margin, while the constraint (4.26) can still be satisfied. One would like to find $\boldsymbol{x}_\mathrm{j}^*$ such that the inequality in (4.26) holds (or violated as slightly as possible) over variations in $\boldsymbol{w}$ and b. For the majority of distributions which are used in real-life problems, the following criterion can be used to obtain the inverse projection $\boldsymbol{x}_j^*$ of the χ_j:

$$\begin{aligned} &\boldsymbol{x}_j^* = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}|\boldsymbol{x}_j, \boldsymbol{r}_j), \\ &s.t.\ f(\boldsymbol{x}) = c_\eta^j. \end{aligned} \tag{4.38}$$

The reasoning behind this criterion is as follows: if $\boldsymbol{x}_j^*$ is fixed at the maximum of $p(\boldsymbol{x}|\boldsymbol{x}_j,\boldsymbol{r}_j)$ at the surface $f(\boldsymbol{x})= c_\eta^j$, then the integral in the left part of (4.26) is less likely to change. Problem (4.38) is a constrained optimization problem, which has to be solved. It results in the desired inverse projections $\boldsymbol{x}_j^*$ which form the new training set. The standard SVM solution for the obtained training set approximates the solution of the initial problem (4.25).

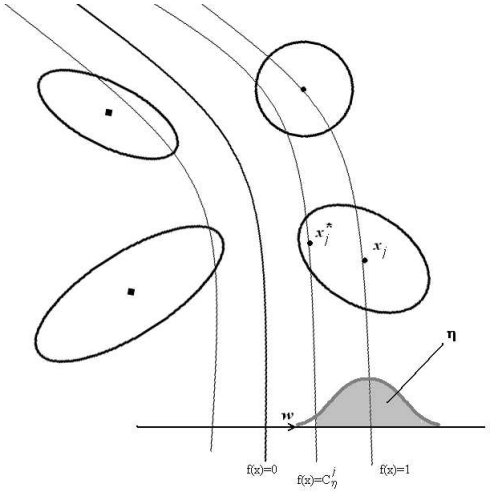


Figure 4.1. The illustration of the hyper-plane projection method. Refer to the text for the notations.

4.3.4 Discrimination of Gaussian Distributions

For the particular case of Gaussian distributions, $p(\boldsymbol{x}|\boldsymbol{x}_i,\boldsymbol{r}_i) = \mathcal{N}(\boldsymbol{x}_i,\Sigma_i)$, the presented scheme results in the following algorithm. The exact linear optimization problem reduces to:

$$\begin{aligned}
&\min \tfrac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{L} \alpha_i (1 - \operatorname{erf} \tfrac{y_i(\boldsymbol{w}\boldsymbol{x}_i+b)-1}{\sqrt{2\boldsymbol{w}^T\Sigma_i^{-1}\boldsymbol{w}}}), \qquad (4.39)\\
&\sum_{i=1}^{L} y_i\alpha_i \left(\boldsymbol{w}^T\Sigma_i^{-1}\boldsymbol{w}\right)^{-\frac{1}{2}} \exp(-\tfrac{(y_i(\boldsymbol{w}\boldsymbol{x}_i+b)-1)^2}{2\boldsymbol{w}^T\Sigma_i^{-1}\boldsymbol{w}}) = 0,\\
&\boldsymbol{w} = \sum_{i=1}^{L} y_i\alpha_i\boldsymbol{x}_i^\star \left(\boldsymbol{w}^T\Sigma_i^{-1}\boldsymbol{w}\right)^{-\frac{1}{2}} \exp(-\tfrac{(y_i(\boldsymbol{w}\boldsymbol{x}_i+b)-1)^2}{2\boldsymbol{w}^T\Sigma_i^{-1}\boldsymbol{w}}).
\end{aligned}$$

Note that the term in the exponent is a Mahalanobis distance from $\boldsymbol{x}_\text{i}$ to the line $y_i(\boldsymbol{w}\boldsymbol{x}_i+b)=1$. This formulation can be used to control the precision of the approximate solutions. We now present the non-linear version of the algorithm using the Gaussian isotropic RBF kernel with parameter δ. The method of hyper-plane projections can be applied using the follow-

ing results for the direct step:

$$\begin{aligned}\mu_j &= \sum_{i=1}^{L} y_i \alpha_i D(\boldsymbol{x}_j, \boldsymbol{x}_i) + b, \\ \sigma_j^2 &= \sum_{i,k=1}^{L} y_i y_k \alpha_i \alpha_k M(\boldsymbol{x}_j, \boldsymbol{x}_i, \boldsymbol{x}_k) - \mu_j^2,\end{aligned} \tag{4.40}$$

where

$$\begin{aligned}&D(\boldsymbol{x}_k, \boldsymbol{x}_i) = \left|\Sigma_i^{-1} + \delta\right|^{-\frac{1}{2}} \cdot \exp(-\frac{1}{2}(\boldsymbol{x}_k - \boldsymbol{x}_i)^T \delta \Sigma_i^{-1} (\Sigma_i^{-1} + \delta)^{-1} (\boldsymbol{x}_k - \boldsymbol{x}_i)), \\ &M(\boldsymbol{x}_k, \boldsymbol{x}_i, \boldsymbol{x}_j) = \left|\Sigma_i^{-1}\Sigma_j^{-1} + \delta(\Sigma_i^{-1} + \Sigma_j^{-1})\right|^{-\frac{1}{2}} \cdot \exp(-\frac{1}{2}(\boldsymbol{x}_k - \boldsymbol{x}_i)^T \Omega (\boldsymbol{x}_k - \boldsymbol{x}_i)), \\ &\text{where} \Omega = \delta\Sigma_i^{-1}((\Sigma_i^{-1}\Sigma_k^{-1})^{-1}(\Sigma_i^{-1} + \Sigma_k^{-1}) + \delta)^{-1}.\end{aligned} \tag{4.41}$$

The inverse projection can then be carried out by solving the following optimization problem:

$$\begin{aligned}&\boldsymbol{x}_j^* = \arg\min_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{x}_j)^T \Sigma_j^{-1} (\boldsymbol{x} - \boldsymbol{x}_j), \\ &s.t. \sum_{i=1}^{L} y_i \alpha_i \exp(-\delta(\boldsymbol{x} - \boldsymbol{x}_i)^2) + b = c_\eta^j.\end{aligned} \tag{4.42}$$

This problem has the following approximate analytical solution:

$$\boldsymbol{x}_j^* = (I + 2\gamma\delta c_\eta^j \Sigma_i)^{-1} (\boldsymbol{x}_j + 2\gamma\delta \sum_{i=1}^{L} y_i \alpha_i \exp(-\delta(\boldsymbol{x}_j - \boldsymbol{x}_i)^2) \Sigma_i \boldsymbol{x}_i), \tag{4.43}$$

for some positive constant γ, which has to be chosen to satisfy the constraint in (4.42).

Despite the cumbersome expressions above, the real computations are significantly simplified, since for high-dimensional input data diagonal covariance matrices are often used.

Links to Related Methods

The most related methods for the particular case of Gaussian distributions discrimination were proposed recently by [3] and [4] and deal with uncertain data. The training samples are considered to be given with some uncertainty, presented in a form of Gaussian distributions.

The method of [3], which is aimed at classifing datasets with missing (uncertain) samples, considers margin maximization under the following constraints:

$$Pr[y_i(\boldsymbol{w}\boldsymbol{x}_i + b) \geq 1 - \xi_i] \geq \eta, \ \xi_i \geq 0. \tag{4.44}$$

As one can see, this formulation is similar to the constraint (4.26). The relevance can be shown by appling the Chebyshev inequality in order to obtain the corresponding deterministic constraint from (4.44).

[4] instead deal with the constraint

$$Pr[y_i(\boldsymbol{w}\boldsymbol{x}_i + b) \geq 1 - \xi_i] \leq \eta, \ \xi_i \geq 0. \tag{4.45}$$

This type of approach leads to more robust models, since, in the end, the less certain samples obtain the least weights. However, this intuition is hardly applicable for the problems

we are aiming to. It was developed for a specific medical applications considered by the authors.

Despite these differences, both models lead to a Second Order Cone Programming (SOCP) optimization problem. This optimization problem can be solved numerically by Interior Point methods [45], which are, however, quite costly in terms of computational time. The proposed approximate approach involves standard SVM QP optimization only.

Moreover, due to the computational costs of the SOCP, both papers mention the need of an approximate solution. The approximation is also based on modifying the means. However, the update rule of [4] is very straightforward, as it suggests updating the samples along the $\boldsymbol{w}$ without taking into account any information on the covariance of the parent distribution.

The related update rule can be easily derived from the approximate formulation proposed by [3]. This, however, was not done by the authors. Nevertheless, let us note that for the linear case of (4.42) the exact solution of the inverse projection for the Gaussians is given by

$$\boldsymbol{x}_j^* = \boldsymbol{x}_j + \frac{c_\eta^j - (\boldsymbol{w}\boldsymbol{x}_j + b)}{\boldsymbol{w}^T \Sigma_j \boldsymbol{w}} \Sigma_j \boldsymbol{w}. \tag{4.46}$$

This linear case almost perfectly coincides with the result which could be derived from the formulation considered by [3].

The significant difference with our method lies in the way the algorithms were kernelized. Non-linearity through the kernel trick is introduced under a number of assumptions in all the methods. It is not possible to "kernelize" the initial algorithms directly. This is also the case for the general original problem (4.31)-(4.34), presented above (and for the particular case of Gaussian distributions as well). However, the developed approximate procedure deals with precise kernels directly by using the feature space of the original SVM for making projections, hence this drawback is partly avoided.

Finally, as it was mentioned above, a number of papers devoted to invariant learning are based on discriminating different objects in the input space. Methods aimed at direct margin maximization were recently proposed by [21], [18]. Furthermore, a general method for defining an SVM kernel function for pairs of distributions was presented by [36].

Concerning the representation of invariances through distributions, a related work was presented previously in Chapter 3. There, we introduced a kernel for standard SVM model, which counts a pair-wise overlap of distributions. This method is subject to the curse of dimensionality. The problem has been avoided in the current approach by introducing the margin between distributions directly.

4.3.5 Experiments

As mentioned in the introduction, there is a broad field of applications for the proposed approach. This includes problems from speech processing, biometric client identification, object categorization, etc.

However, the presented and some related approaches can also be used for handling invariances in pattern recognition problems. A similar setting can also be applied for the task where data samples are given with some uncertainty of a known nature. Here, we present experiments on invariant face image classification.

Let us start with a simple synthesized 2-D data example, which nicely illustrates the presented approach.

Toy Data Experiments

The task to be solved is a 2-class classification of 2-D Gaussians (see Figure 4.2). The dataset contains several samples of each class; their covariances are illustrated by ellipses around the means. The modified training set of $\{\boldsymbol{x}_-^*\}$ is shown with black dots, which form a curve for different value of γ in Eq.4.43. The samples $\boldsymbol{x}_j^*$ coincide with the means $\boldsymbol{x}_j$ for $\gamma = 0$, and tend to the decision boundary according to the covariance of their parent distribution, as γ increases. Final samples (shown by filled boxes) correspond to the $\eta = 0.9$. The modified decision boundary $f^*(\boldsymbol{x}) = 0$ is shown with a thick line.

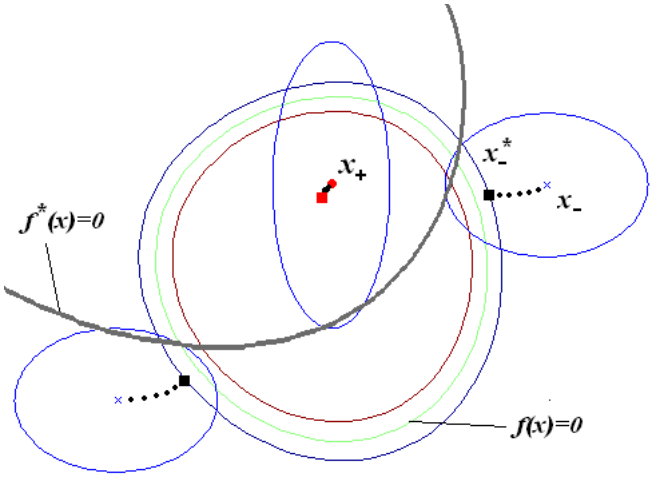


Figure 4.2. Toy Data Discrimination. The decision boundary changes according to the covariances of the distributions in the training set.

Invariant Image Classification

The task of invariant image classification is still a challenging problem in computer vision. While a number of approaches for dealing with simple images like Optical Characters exist, methods for complex real-life images are still under development.

A natural way to model invariances is to consider how the desired transformation change the input samples. Generally, this dependence is very complex and highly non linear, hence difficult to model. Linear approximations are thus used instead. We use tangent vector approximation, as it was introduced in the previous Chapter.

Let ℓ_i^j denotes the tangent vector, which corresponds to the j^{th} invariant transformation of the sample $\boldsymbol{x}_i$.

Table 4.1. Classification accuracies for the compared algorithms on face image classification problem.

ALGORITHM	TR.ERR.,%	TEST.ERR.,%	TIME, S
SVM	0.5	11.2	0.75
VSV SVM	0.4	9.9	9.6
TVK SVM	0.5	9.7	4.2
SVM GAUSS	0.5	9.7	2.8

The corresponding tangent vector based Gaussian distribution is as follows:

$$P(\boldsymbol{x}|\boldsymbol{x}_i, \{\ell_i^1, \dots \ell_i^J\}) = \frac{\exp(-(\boldsymbol{x}-\boldsymbol{x}_i)^T L_{x_i}^{-1}(\boldsymbol{x}-\boldsymbol{x}_i))}{(2\pi)^{N/2}|L_x|^{1/2}}, \quad \text{where } L_x^{-1} = \sum_{j=1}^{J} \frac{\ell_i^j \ell_i^{j\,T}}{2\gamma_j^2 \ell_i^{j\,2}}. \tag{4.47}$$

We consider these distributions as a training set for our algorithm. Parameter γ_j controls the effective width of the distribution for the given direction of ℓ_i^j. It can be fixed to some value according to some prior knowledge, since the resulting transformed images can be visualized. We used the same value of γ for all the invariances. Generally, the resulting Gaussians have full-ranked covariances, that dramatically slows down the overall computations.

Face Data Classification

We conducted experiments using images of the faces detected from movie scenes using a face detector, described in [64]. There is a total of 2899 images in the database. The data is available at [http://www.robots.ox.ac.uk/∼vgg/data]. This data were used in the previous chapter, and a (Distribution-based) Tangent Vector Kernel method was applied to the task of binary face image classification. Example images and their corresponding labels are presented in Figure 3.2. As it is mentioned, that approach still suffers from a curse of dimensionality problem.

Two basic invariant transformations were considered: scalings and rotations. Finite difference vectors, obtained as a difference of an original and a transformed images were used instead of the original tangent vectors. The reason is that real tangent vectors fail for such complex and non-smooth images as faces when the transformation is more than infinitely small. Different tangent vectors were used for the rotations to the left and right, as well as for the zoom in and zoom out scalings, as it was described before in Section 3.4.4.

The parameters of the algorithms were chosen according to the minimum of cross-validation error over the training set, resulting in the following values: $\delta = 2{\cdot}10^{-5}$, $C = 100$, $\gamma = 1000$. Table 4.1 presents training and testing errors obtained with SVM with Gaussian RBF kernel (SVM), SVM trained with virtual samples (VSV SVM), SVM with Tangent Vector Kernel (TVK SVM) [50] (explained in the previous chapter), and the developed method (SVM Gauss). We consider the Virtual SV method as a state-of-the-art approach to invariant learning with SVM-based methods. Given unlimited computational resources, this is currently the method of practical choice.

While both methods are statistically significantly better than baseline SVM (with 95% confidence), no significant improvement was observed in comparison with Virtual Support Vectors in terms of the testing error. However, the proposed approach is faster in terms of

operational time. This advantage is even more significant if the covariance matrices are diagonal.

4.3.6 Discussion and Conclusions

While classical SVMs discriminate between example points of two classes, we proposed in this chapter a novel SVM formulation to discriminate between example distributions of two classes, while still keeping advantages of SVMs such as margin maximization and kernel trick. This presentation is based on the publications [54], [53]. The devloped extension can be used for many different settings, including principled incorporation of invariances described by distributions, which was illustrated with experiments in Section 4.3.5. Other possible uses of this model include the possibility to maximize the margin for problems that were traditionally solved by generative models and log likelihood ratios such as speech processing.

Since the direct solution was not tractable, we presented an approach for an approximate solution of the optimization problem. This approach consists of two simple projection steps, resulting in a modified training set. Thus, possible problems with the convergence of an iterative scheme are avoided. Next, the algorithm was turned into a nonlinear version through the usual "kernel trick". The feature space of the original SVM (trained on the means of the distributions) is exploited for the latter. The algorithm demands a standard SVM QP solver only. The case of Gaussian distributions was considered in details, and some links to related research were provided.

The algorithm was applied to the real problem of invariant face image classification. The knowledge on invariances was incorporated into the algorithm by considering a special type of distributions, based on tangent vectors. The comparison to the state-of-the-art virtual support vector was provided. Currently, the method is found to be competitive with the state-of-the-art, and the proposed solution was preferable in terms of computational time.

Concerning invariant learning problems, the basic advantage of the proposed method is that it maximizes the margin between invariance-modelling distributions directly. Note that approaches based on measuring the pair-wise overlap between distributions are subject to the curse of dimensionality if the linear approximation to the invariant transformations is used.

For a practitioner, the algorithm provides a nice feedback. As $\boldsymbol{x}_j^*$ are known, these samples can be visualized. An interesting question is whether these samples coincide with human's intuition to be the most discriminative. To our knowledge, the answer is positive most of the times.

Up to now, we have considered a number of approaches which use linear approximation of the desired invariant transformation. Whether the kernel or the algorithm was modified to include this knowledge, the manifold was considered to be linear. Moreover, the transformation was often assumed to be differentiable. In real life, this is not the case. Often, the desired transformation is complex and is hardly representable analytically. While processing real-life images, even small transformations (such as considered rotations and scalings) lead to lighting conditions changes, background changes, etc. It also concerns the cases when feature selection is carried out before the classification step, since feature

selection is often a complex non-linear transformation. In these cases, the only available information about the manifold is actually the set of samples which belong to it. The universal virtual sample method remains the only one to handle invariances in this situation. It may lead to the very large-scale optimization problems.

In the next chapter, we consider a method which handles invariances in the described situation, when the only information given is the set of samples belonging to the transformation manifold. It is based on kernel modification and does not require enlarging the training set with virtual samples.

Chapter 5

Prior Knowledge from Unlabelled Data

The idea of using the inner geometric structure of the data for better data processing algorithms is of increasing attention in Machine Learning. This trend arises from unsupervised methods such as clustering and dimensionality reduction techniques. A clever use of geometrical structure should improve the performance of any learning algorithm. In particular, semi-supervised methods are under rapid development recently [14]. These methods exploit unlabeled data, i.e. those data samples which consist of the input values only, while the desired output value is unknown. In fact, most real-life learning problems are actually semi-supervised. For example, this is the situation when a huge amount of images are available, but only a part of them is annotated, i.e. labeled.

The information one obtains from the unlabeled part of the dataset can be of different nature. A reasonable assumption to make is the following. Assume the data lies on some lower-dimensional manifold in the original input space. Using some properties of the manifold, data analysis methods can be improved, as shown in recent developments devoted to the exploration of such an approach (see [40], [1] and references therein for instance). Furthermore, given the explosive growth of interest in the field of kernel methods, non-parametric data-dependent kernels which reflect the inner geometry of the data are of particular interest. A general approach was recently proposed in [68].

In this chapter, which is based on the publications [51], [52], we apply this framework to approach the main problem of this research, namely prior knowledge incorporation. Concerning classification, the methodology developed for semi-supervised learning is adapted to model the manifolds induced by the desired invariant transformations. The kernel is constructed in a way to produce smooth decision functions on the modeled invariant manifolds, therefore preserving the class membership on these manifolds. Afterward, a kernel classifier is applied to the task. For regression, we implement this kernel to enforce the model to be smooth on the support of the estimated function, or, in other words, on the manifold which is modelled using unlabelled data.

We introduce manifold learning in Section 5.1, present the way to adapt this framework to invariant learning in Section 5.2, and introduce the corresponding kernels in Section 5.2.2. We provide some practical issues and experimental results on a real Optical Character Recognition (OCR) task in Section 5.3. Next, we provide the implementation

of a similar technique for regression estimation in Section 5.4, and present experimental results on regression estimation in Section 5.6. We conclude the chapter in Section 5.7.

5.1 Learning on Manifolds

As it is introduced in Machine Learning, the supervised learner aims at estimating the input-output relationship (dependency or function) $f(\boldsymbol{x})$ by using a training data set $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, N$ where the inputs $\boldsymbol{x}$ are n-dimensional vectors and the labels (or system responses) y are continuous values for regression tasks and discrete (e.g., boolean) for classification problems.

However, the situation where some labeled patterns are provided together with unlabeled ones, arises frequently. This is called *Semi-Supervised Learning*. Currently, this is an actively developing field [14]. When predictions have to be made to given unlabeled locations only, this particular situation is called transductive learning [74], [9].

Recently several approaches to semi-supervised learning were proposed. Low Density Separation (LDS) algorithms [13], Transductive SVMs, Graph and Gradient Transductive SVMs [31], and a group of Manifold Learning methods [1] are the core of those recently developed techniques.

Here we give a basic idea for the last group of methods, namely, Manifold Learning. The so-called *manifold assumption* is accepted in this framework. This implies that data actually belong to some lower dimensional manifold in high dimensional input space. Thus, it is reasonable to build models which exploit regularization on the manifolds.

Usually the only information about the manifold is the finite set of (unlabeled) samples, $\{\boldsymbol{x}_i\}$, $i = N+1, \ldots, M$. Thus, the model has to be smooth (regularized) on the corresponding graph, whose nodes are data samples and edges are constructed to preserve the geometrical properties (geodesic distances) on the graph. Let an edge connecting $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ have some weight w_{ij}; zero value means that the nodes are not connected.

A nice property here arises from the notion of graph Laplacian. It is defined as

$$\mathcal{L} = D - W \tag{5.1}$$

where W is the matrix with elements $w_{ij} = \exp(-\delta(\boldsymbol{x}_i - \boldsymbol{x}_j)^2)$, δ is some positive constant (usually proportional to the avarage length of the graph edges), and D is a diagonal matrix with $d_{ii} = \sum_j w_{ij}$.

It can be shown that eigenvectors of $\mathcal{L}$ provide a natural basis on the graph, giving rise to regularization by penalizing the complexity. This can be done by minimizing the norm in the space of functions defined on graphs. Please refer to [1] for details and solid justification behind this technique.

5.2 Graph Models of Invariant Manifolds

Now, we present an approach for applying the technique of modeling data transformation manifolds for invariant learning with kernel methods. The approach is based on building a kernel function on the graph modeling the invariant manifold. It provides a way for taking into account nearly arbitrary transformations of the input samples. The approach

is verified experimentally on the task of optical character recognition, providing state-of-the-art performance on harder problem settings.

One of the well-known approaches to invariant learning is the Tangent Distance method [67]. As we described before, it proposes to replace the Euclidean distance between data samples with a distance between the corresponding linear tangent manifolds defined by tangent vectors of the desired invariance transformation. This method was successfully applied to Optical Character Recognition (OCR) tasks. Its direct application for defining a kernel for SVMs was studied in [25]. A restriction of these methods is that they are suited for distance-based kernels only. The proposed method, on the other hand, does not suffer this restriction.

The decision function, which is smooth on the corresponding manifold, provides invariant classification. This smoothness guarantees that the decision for class membership is unchanged as one considers samples from the invariant manifold (i.e. transformed samples). There is a justification behind this intuition, which is explored, for instance, in [37], and was mentioned in Chapter 2, Section 2.4.1. Moreover, as it is shown in [37], this property is closely linked to the type of regularizer used.

The above-mentioned direct approach to enforce smoothness in the direction of tangent vectors is considered in [11]. This method, however, leads to complicated optimization and appeared to be impractical in real-life tasks.

5.2.1 Graph-based Invariant Manifolds

Given a training sample $\boldsymbol{x}_i$, consider a set of corresponding virtual samples, generated by applying the desired (and, virtually, arbitrary) transformation $G(\boldsymbol{x}, \alpha)$:

$$\{\boldsymbol{x}_i^k\} = G(\boldsymbol{x}_i, \alpha),\ \alpha \in \Lambda, \tag{5.2}$$

where α is a vector of parameters from some finite set Λ (such as a set of rotation angles). Then, a graph is built for every training sample by connecting and setting weights for the nodes $\boldsymbol{x}_i^k$ sharing the same original sample $\boldsymbol{x}_i$. The weights w_{ij} are set to $\exp(-\delta(\boldsymbol{x}_i - \boldsymbol{x}_j)^2)$ if nodes are connected, zero otherwise. Considering the graph-based manifold models and enforcing smoothness of the model on the graph, we constrain it to be invariant to the transformation which generated the manifold.

Next, we introduce a kernel, adapted from [68], to apply a kernel classifier such as Support Vector Machine to graph based manifolds.

5.2.2 Kernels For Invariant Manifolds

Hereafter, we briefly present a method, adapted from [68] for constructing non-parametric semi-supervised kernels which deals with graph-modeled invariant manifolds. As a reminder, this implies that it corresponds to a dot product in some space (Reproducing Kernel Hilbert Space, RKHS), sometimes referred to as a feature space. Generally, the choice of the kernel function is an open issue.

We will follow the notation of [68]. Given data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, and some RKHS H, consider the evaluation map $S(\mathbf{f}) = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)); S : H \rightarrow \mathbf{R}^n$. The semi-norm on $\mathbf{R}^n$ is given by a symmetric semi-definite matrix $\mathbf{M}$,

$$\|S(\mathbf{f})\|^2 = \mathbf{f}^T \mathbf{M} \mathbf{f}, \tag{5.3}$$

where we denoted $\mathbf{f} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))$ and T means transpose.

The exact explicit form of the corresponding reproducing kernel $\tilde{k}(\boldsymbol{x}, \boldsymbol{x}')$ was derived in [68] and is given by:

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - k_{\boldsymbol{x}}^T (\mathbf{I} + \mathbf{MK})^{-1} \mathbf{M} k_{\boldsymbol{x}'} \tag{5.4}$$

where $\mathbf{K}$ is the complete kernel matrix of $k(\cdot, \cdot)$, $k_{\boldsymbol{x}}$ represents one column of $\mathbf{K}$ and $\mathbf{I}$ is the identity matrix. In the presence of unlabeled data, the choice of $\mathbf{M}$ implements the smoothness assumption with respect to its geometric structure. As shown in [1], this is achieved by taking $\mathbf{M} = \gamma\mathcal{L}$, $\mathcal{L}$ being the Laplacian matrix of the graph built on unlabeled samples, and γ a regularization parameter which defines the extent of kernel deformation. By setting γ=0 one obtains the original kernel, as it is clearly seen with (5.4), and no invariance information is used in the model.

This kernel can be plugged into any kernel classification algorithm. We will use the widely known standard form of soft margin SVMs, see Section 2.2.1. This method provides a classifier by taking the sign of the kernel expansion:

$$f(\boldsymbol{x}) = \left(\sum_{i=1}^{N} \alpha_i \tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}) + b \right), \tag{5.5}$$

where the weights α_i are obtained from solving the QP optimization problem (2.21).

The advantage of the method in computational speed is that the size of this QP is the same as the size of the original optimization problem. In virtual samples methods, however, the training set size and, correspondingly, the optimization problem dimension is increased [60]. However, each kernel computation becomes more expensive. The main idea to obtain improved performance of kernel computation is based on the observation that graph Laplacian has block-diagonal structure. It open perspectives to speed up the processing. Roughly, given N original training samples and L virtual ones, used as graph nodes, the proposed method requires $O(N^3 + NL^3)$ computations instead of $O(N^3L^3)$ as requires SVM with virtual samples.

5.3 Classification Experiments

The experiments described below deal with global rotational invariance as an example. We start with an illustration of modeling the invariant manifold and kernel construction with graphs, using character images. Finally, we test our approach on a real-world handwritten digits dataset, commonly used in machine learning for benchmarking different algorithms and known as USPS digits.

5.3.1 Practical Issues

We first start with a discussion on some issues which arise while implementing the described method.

Manifold modeling. There are two basics to take into account while constructing the graph: the smoothness of the transformation, which is modeled as locally linear between the adjacent nodes; and the number of nodes, which has to be sufficient to model the manifold reliably.

A workaround is to build the graph by generating a sufficient number of virtual samples as nodes, and connect the K nearest neighbors. The rest of the procedure remains unchanged. This approach will capture some intra-class similarity in the data. However, the influence of noise in data such as mislabeling or outliers will probably be increased.

Choice of parameters. There are several parameters which influence the final classification model. A general way to tune them would be to carry out cross-validation on the training data. However, this is complicated since the parameter space is of high dimension. Here we consider several heuristics to simplify the choice.

There are two groups of parameters. The first group corresponds to the manifold-modeling graph. These are δ and γ. The parameter δ is taken such that the bandwidth of RBF function in graph Laplacian is equal to the average distance between the graph nodes. The influence of the γ parameter is explored experimentally below.

The second group contains the hyper-parameters of the learning algorithm, which include the trade-off parameter C of the SVM and the kernel parameters. In this paper, we used the kernel described in Section 5.2.2, using the standard Gaussian RBF kernel with bandwidth σ as a base kernel. These parameters are tuned using the standard cross-validation technique on the training data.

Features. The only restriction for the feature space used is to guarantee the smoothness of the manifold in the feature space. For the sake of simplicity and to provide an easy way for visualization, we use the pixel intensities of the raw images as inputs in the experiments below. However, the use of this kernel in the feature space representation is an important issue for future developments.

5.3.2 Global Rotation

The purpose of this section is to provide empirical evidence for the method, and particularly manifold modeling. We consider the problem of kernel construction for character image classification.

Figure 5.1 presents a contour plot of the kernel function centered at image A. Since the basic kernel is an RBF one, this value can be considered as a measure of similarity. The angle at the polar plane corresponds to the rotation angle of the image, and radius corresponds to the lag of vertical translation of the image before rotation. Black dots are the unlabeled rotated images used as graph nodes.

Figure 5.2 presents the value of kernel function centered at the original image of the letter, as a function of the angle the image was rotated. The values were normalized. As one can see, the parameter γ controls the amount of invariance information introduced by virtual inputs.

5.3.3 USPS digits

In this section we carry out experiments on a widely used benchmark: the USPS dataset of handwritten digits. It consists of 7291 training and 2007 testing samples. These are grey scaled images of 16x16 pixels.

The data were modified by applying rotation to each character image. The rotation angle is random in the range 0-360 degrees (see Figure 5.3). In this modified setting, this classification problem becomes extremely difficult. The proposed algorithm was applied to

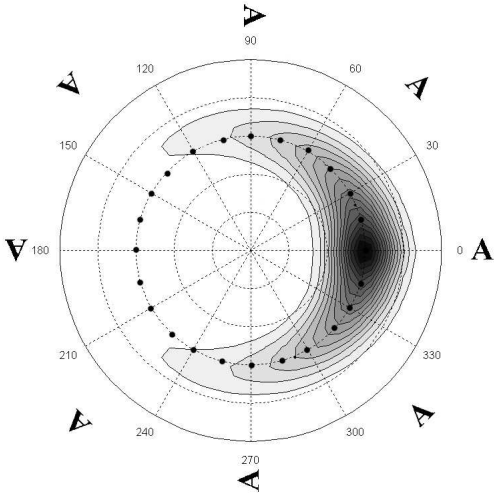


Figure 5.1. Kernel centered at image 'A'. The value of the kernel function can be considered as a similarity measure.

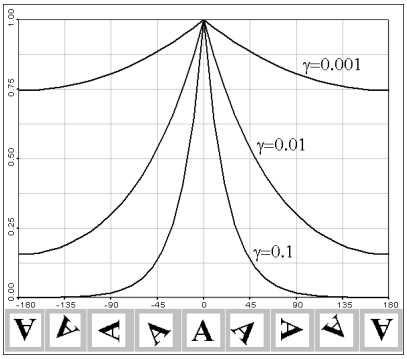


Figure 5.2. Kernels for different γ values. This parameter controls the amount of invariance information incorporated into the kernel.

binary classification. The task was to classify digits "$0-4$" against "$5-9$". Graph nodes were constructed by consequently rotating the original images on 15 degrees. This results in 23 virtual samples per image. The training set was split into 36 subsets of 200 samples. The results averaged over splits are presented in Table 5.1. Standard SVM obviously fails to classify the rotated digits.

Another method which was tested at this experimental setting is a regularized least squares classifier (RLS). The decision function (5.5) is used in this method. The weight α are estimated by minimizing the empirical risk (training error) with quadratic regularization term. The method is faster then SVM-like methods since this is an unconstrained optimization solved by gradient descent.

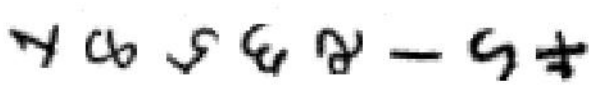


Figure 5.3. Some rotated USPS digits.

Table 5.1. Testing errors on USPS data. Standard SVM fails on rotated data. Methods with graph-based kernels outperform the virtual samples methods in computational time providing competitive generalization performance.

	Algorithm	Testing Error, %	Time, s
	SVM (unrotated data)	12.0	8.5
	SVM (rotated data)	34.0	8.5
	VSV SVM	13.1	2160
	Graph SVM	12.5	480
	RLS	14.0	1150
	Graph RLS	13.2	280

5.4 Kernel Regression with Unlabelled Data

This section presents a semi-supervised kernel method for regression estimation in the presence of unlabeled patterns. We incorporate the prior knowledge about the support of the function. The method exploits a recently proposed data-dependent kernel which is constructed in order to represent the inner geometry of the data. This kernel is implemented into Kernel Regression methods (SVR, KRR). Experimental results aim to highlight the properties of the method and its advantages as compared to fully supervised approaches. The influence of the parameters on the model properties will be evaluated experimentally. One artificial and two real-world datasets were used to demonstrate the performance of the proposed algorithm.

5.4.1 Manifold Regularization

Most of the work done in the field of semi-supervised learning field is related to fully unsupervised tasks or semi-supervised classification problems. Semi-supervised regression methods are, however, much less studied. The research done in this direction include [1], [28], [14]. In this section, we combine recent developments in the field of manifold learning with kernel regression learners such as Support Vector Regression and Kernel Ridge Regression.

5.5 Kernel Regression Methods

The same kernel as presented above is implemented into kernel regression methods. The exact explicit form of the kernel $\tilde{k}(\boldsymbol{x}, \boldsymbol{x}')$ is given by:

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - k_{\boldsymbol{x}}^T (\mathbf{I} + \mathbf{MK})^{-1} \mathbf{M} k_{\boldsymbol{x}'} \tag{5.6}$$

where $\mathbf{K}$ is the complete kernel matrix of $k(\cdot,\cdot)$, $k_{\boldsymbol{x}}$ represents one column of $\mathbf{K}$ and $\mathbf{I}$ is the identity matrix. In the presence of unlabeled data, the choice of $\mathbf{M}$ implements the smoothness assumption with respect to its geometric structure. As shown in [1], this is achieved by taking $\mathbf{M} = \gamma\mathcal{L}$, $\mathcal{L}$ being the Laplacian matrix of the graph built on unlabeled samples, and γ a regularization parameter which defines the extent of kernel deformation. By setting γ=0 one obtains the original kernel, as it is clearly seen with (5.6).

The first method which we will use in the experiments below is the standard form of Support Vector Regression, as presented in Section 2.2.2. The second one is the Kernel Ridge Regression.

5.5.1 Kernel Ridge Regression

Kernel Ridge Regression is a regularized least square approach, which leads to the same form of regression function (5.5). However, it exploits the square loss function, and α coefficients can be obtained from the following closed form expression:

$$\alpha = (\mathbf{K}^T\mathbf{K} + \delta\mathbf{I})^{-1}\mathbf{K}^T\mathbf{Y} \tag{5.7}$$

where δ is a regularization parameter and $\mathbf{Y}$ is the vector of training outputs. Note that an iterative method can also be used to train the KRR model.

5.6 Regression Experiments

The experiments described below were carried out on the following datasets: spiral, Boston housing and sunspots. The first one is an artificial dataset, which we use to explore and illustrate the basic properties of the method. The other two are real-world datasets, commonly used in machine learning for benchmarking different algorithms.

5.6.1 Kernel Choice

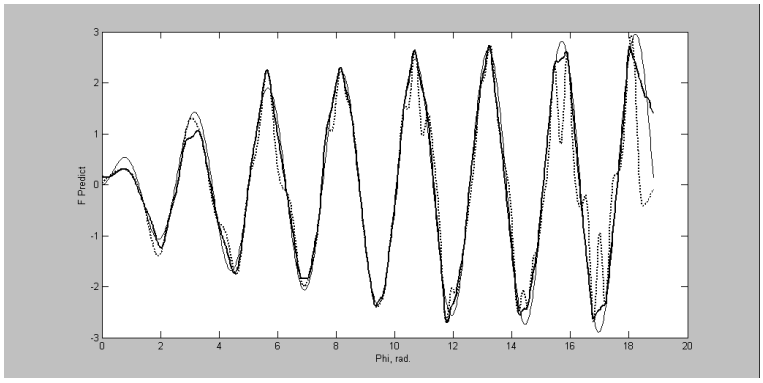
In these experiments, we used the kernel described in Section 5.5, using the standard Gaussian RBF kernel with bandwidth σ as a base kernel. Gaussian RBF is used in all the baseline supervised algorithms as well. Another parameter to select is the regularization parameter γ of the modified kernel (5.4). This will be explored empirically in this section.

5.6.2 Spiral: 2D Synthetic Example

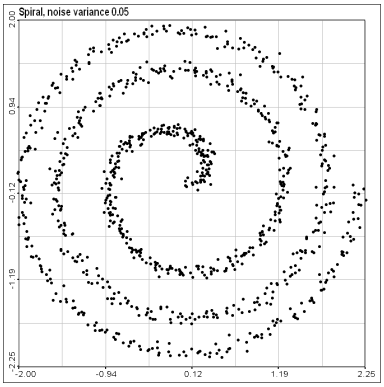
This dataset was artificially generated with:

$$\begin{cases} x^1(\phi) = \frac{1}{2}\sqrt{\phi}\cos(\phi) + N(0,\sigma_x) \\ x^2(\phi) = \frac{1}{2}\sqrt{\phi}\sin(\phi) + N(0,\sigma_x) \\ f(\phi) = \ln(1+\phi)\sin(\frac{5}{2}\phi) + N(0,\sigma_f) \end{cases} \tag{5.8}$$

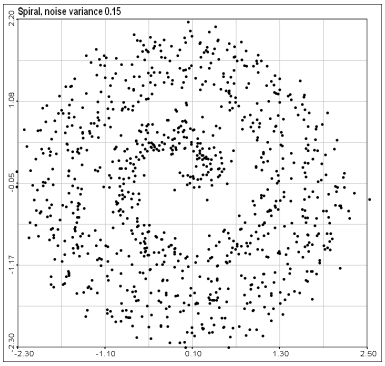
in the range of $\phi \in [0; 6\pi]$. The function $f(\phi)$ to predict is defined on the 2D spiral. This function is presented in Figure 5.4a with a thin solid line. Both coordinates and function values are corrupted with normal noise of variance σ_x^2 and σ_f^2 correspondingly. Two random data realizations are presented in Figures 5.4b and 5.4c. We compare the performance of



(a) Function $f(\phi)$ and its estimates by semi-supervised (bold line) and standard (dotted line) SVR.



(b) Input samples, σ_x=0.05



(c) Input samples, σ_x=0.15

Figure 5.4. 2D spiral data used for method validation.

the proposed method and the standard Support Vector Regression with Gaussian RBF kernel. Labeled part of the training set consist of 100 randomly selected samples, while an other set of 900 samples were provided unlabeled to the semi-supervised method. The results are averaged over 10 runs of the algorithm (each run selecting different training and test examples), and its performance was measured in terms of RMSE using the known underlying function $f(\phi)$ (5.8).

Figure 5.5a presents the dependence of the testing error of both methods with respect to the variance of noise in the inputs σ_x. The top curve (with higher RMSE) corresponds to standard SVR. As can be seen, semi-supervised regression (bottom curve) is preferable for a large region of noise variance, provided that some geometrical structure in the data remains.

Figure 5.5b presents the dependence of the testing error of both methods with respect to the kernel regularization parameter γ. The dashed line corresponds to the testing error of the basic SVR. The semi-supervised method outperforms SVR for a large range of values.

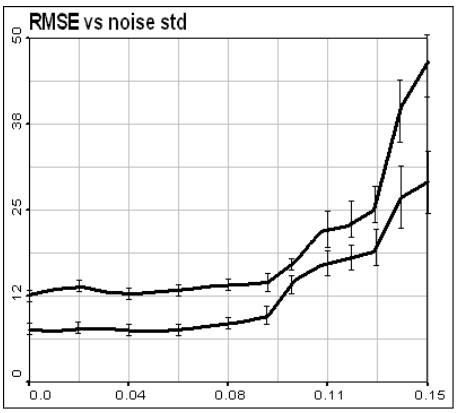
Table 5.2. Experimental results for Boston Housing database.

	Boston housing results		
Algorithm	Train err.	Test err.	Training time, s
SVR	4.0	5.3	0.3
SemiSVR	3.5	5.0	0.5
KRR	2.7	4.0	0.3
SemiKRR	3.5	4.0	0.5

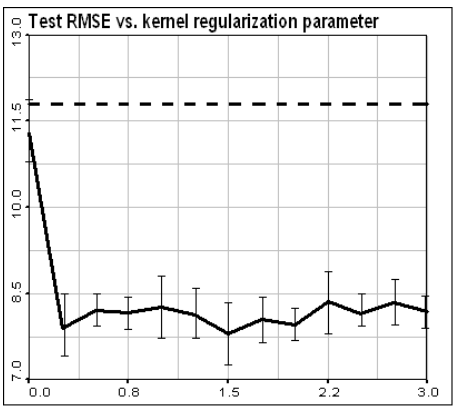
5.6.3 Boston Housing: High Dimensional Regression Estimation

The task here is to predict the median price of the houses in certain area of Boston based on 12 continuous and 1 binary variables defining the characteristics of the area. The training dataset consists of 466 samples, while 40 samples were reserved for testing. The parameters of the methods were tuned with cross-validation error. Unlabeled data were randomly chosen by removing the labels from 50% of data samples. The results were averaged over 10 runs of the algorithm (each run with different training and test examples).

Table 5.2 presents training, testing error and training time of the following algorithms: the considered SVR with semi-supervised kernel (SemiSVR), SVR with Gaussian RBF kernel, the standard method of Kernel Ridge Regression (KRR), and KRR with a semi-supervised kernel (SemiKRR). The results suggest that no significant improvement was achieved on this dataset. There was probably not enough data samples to model the manifold in the 13-dimensional input space.



(a) RMSE, X-axis: σ_x.



(b) RMSE, X-axis: γ.

Figure 5.5. Spiral data experimental results.

5.6.4 Sunspots: Time Series Prediction with Missing Values

This dataset is a time series representing the number of visible sunspots per day. The following embedding was used to apply a regression estimator for predictions: to predict the yearly average of the year starting next day using the previous 12 yearly averages. The series is thus smoothed by averaging. Figure 5.6a presents some 2D projections (trajectories)

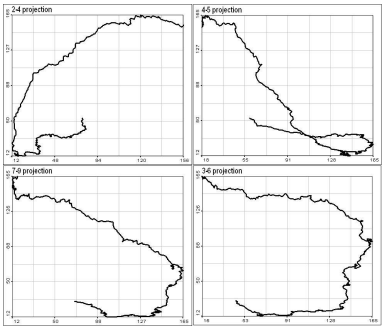
Table 5.3. Experimental results for the Sunspots database.

	Sunspots results		
Algorithm	Train err.	Test err.	Training time, s
SVR	10.3	15.8	10.1
SemiSVR	9.4	12.3	30.4
KRR	11.3	17.5	12.6
SemiKRR	12.1	14.0	35.3

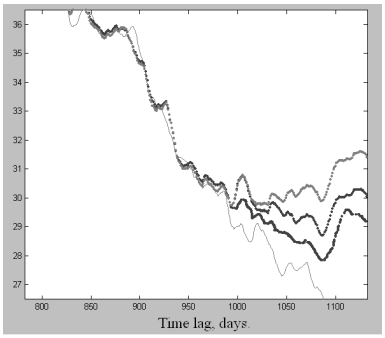
in the embedded input space. One can observe a distinct structure of the inputs, which justifies the use of manifold-based semi-supervised methods for making predictions.

We used only one part of the series containing 2000 values. 50% of the labels were deleted from the series to simulate missing data. Hence, the unlabeled part of the dataset consisted of 1000 samples, and the training set also contained 1000 labeled samples. Missing values in inputs were averaged using two nearest neighbors in time. The obtained results are summarized in Table 5.3. The results are averaged over 10 runs of the algorithm where different sections of 2000 points were selected randomly.

Predictions are presented in Figure 5.6b, for γ=0 (standard SVR), γ=0.1, γ=1. The semi-supervised SVR gives better forecasting for longer time periods, for higher values of γ.



(a) Some 2D trajectories



(b) SemiSVR Predictions

Figure 5.6. Sunspots database results. Semi-supervised SVR provides better predictions.

5.7 Discussion and Conclusions

The universal approach to invariant learning is the virtual samples approach. Given unlimited computational resources and an ability to add enough virtual samples to the training set, all the information on invariances can be learned directly from data.

An alternative to this approach, proposed in this chapter and published in [51], consists in modeling the invariant manifolds instead in order to obtain improvements in training time without lack of precision. We adapted the recently developed method to model manifolds defined by samples and we built a kernel classifier which enforced smoothness on these manifolds. We thus obtained an invariance property of the classifier.

The method provides a way to model nearly arbitrary invariances. It requires additional computations to build the kernel. At the same time, the size of the optimization problem is unchanged. The amount of invariant information used in the algorithm can be tuned by the choice of parameter γ.

Promising classification performance on a real OCR task was observed. Other applications, such as dealing with specific invariances or matching problems are of particular interest for further developments.

Concerning kernel regression methods, we proposed to implement the recently developed data-dependent semi-supervised kernel for regression estimation methods, namely Support Vector Regression and Kernel Ridge Regression. Thus, the methods are adapted for semi-supervised learning problems. Some issues of the practical use of the methods were considered. We have shown that the semi-supervised methods do benefit in the case where there exists some geometrical structure in data. A significant improvements in performance compared to baseline supervised kernel regression estimators was shown in a number of experiments on synthetic and real-life datasets, these results were published in [52].

Chapter 6

Conclusion and Perspectives

6.1 General Summary

Throughout this thesis, we have explored an important problem of learning with kernel methods in the presence of prior knowledge. We have shown that the smart use of the latter improves the performance of the algorithms. Several approaches were proposed to incorporate prior knowledge into kernel learning. Our main interest was in the scope of invariant learning. The task here is to enforce the learning algorithm to obey invariance properties. That is, the output of the learning system must not change if the input is changed by a desired invariant transformation.

Particularly, we developed kernels that incorporate the knowledge about sample transformations. Applying the desired transformation to some training sample, one obtains an object in the input space. Next, a kernel is constructed which deals with objects. A number of implementation techniques of this method, based on hard geometrical objects and soft objects based on distributions were considered. The method was applied to real-life task of face images classification and EEG signals classification. The results of this research were published in [55], [50].

In a number of tasks, prior knowledge can be formulated by replacing the input vectors with distributions. A novel algorithm for classifying distributions by their domains was developed. The algorithm combines the principle of margin maximization and a kernel trick, applied to distributions. Thus, it combines the discriminative power of Support Vector Machines and the well-developed framework of generative models. The algorithm can be applied for introducing some prior knowledge on invariances into a discriminative model. It was verified on face image classification task [54]. Other applications such as object categorization were discussed in [53].

Currently, semi-supervised learning is one of the most important branches of ML. Most of the real-life problems are actually semi-supervised. We considered the group of methods known as manifold learning. Considering regression estimation, we implemented a recently proposed data-dependent kernel which is constructed in order to represent the inner geometry of the data. This kernel was built into kernel regression estimators. Experimental results revealed the properties of the method and its advantages as compared to fully supervised approaches. The results of this study are published in [52]. Concerning classification, we adapted the manifold learning approach to deal with invariant manifolds,

generated by the desired transformations. It is based on building a kernel function for the graph which models the invariant transformation manifold. It provides a way for taking into account nearly arbitrary transformations of the input samples. The method is verified experimentally on the task of optical character recognition [51], providing state-of-the-art performance on harder problem settings.

Several links between existing and newly developed approaches are described in the thesis. Regularization techniques, Vicinal Risk Minimization, Virtual Samples approaches are described and compared to the developed methods. However, there is a room for further developments.

6.2 Possible Future Directions

A number of interesting issues remain open in the described field. Moreover, some directions for further research arose as a result of the obtained achievements.

Considering invariant kernels (Chapter 3 and Chapter 5, classification part) it is interesting to generalize the presented approaches into a unified framework of using models of invariant manifolds based on virtual samples without enlarging the actual size of the optimization problem. It would be interesting to obtain stronger links of these approaches with the regularization framework.

Then, the developments of Chapter 4 (namely, the kernel classifier for distributions) can be applied for a number of problems, where input data can be presented in the form of distributions. Currently, it was applied to invariant image classification. An attention can be paid to other problems such as biometric authentication, speech processing, object categorization. This method can be also applied to speed up SVM training, considering the clusters of data in the input space in the form of distributions.

Concerning the use of graphs for building models of invariant manifolds (Chapter 5), the presented approach opens a number of perspectives for future developments. Given a method for dealing with nearly arbitrary transformations of the input samples, the following research directions are of particular interest. First, real-life applications in computer vision tasks can be considered. Currently, the approach was applied to OCR tasks, and invariant manifolds were modeled in the input space of raw images. The usage of the approach for modeling manifolds in feature spaces (using SIFT and similar invariant features) can be explored. It opens a way for achieving intra-class invariances in scene and object categorization problems, classification of object images taken from different viewpoints and in different lighting conditions, etc. Second, in biometric applications, not only the standard transformations such as scalings, rotations and translations, but such transformations as emotional and age changes can be modeled efficiently. Since the amount of added invariant information can be easily controlled with a single hyper-parameter, the trade-off between discrimination abilities and false acceptance can be tuned. Thus, the performance of the authentication system can be controlled.

Generally, the growing interest of the field of machine learning with kernel methods (both from applied and theoretical areas) opens promising perspectives for many further developments in the field. The increasing number of applications demands for new extensions of the existing algorithms, approaches and implementations.

Acknowledgments

This research has been partially carried out in the framework of the European project LAVA, funded by the Swiss OFES project number 01.0412. It supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, funded in part by the Swiss OFES. It was also partially funded by the Swiss NCCR project (IM)2.

Acronyms

BCI	Brain-Computer Interface
DB-TVK	Distribution-Based Tangent Vector Kernel
EM	Expectation-Maximization
EEG	ElectroEncephaloGram
ERM	Empirical Risk Minimization
GM	Generative Models
GMM	Gaussian Mixture Models
HMM	Hidden Markov Model
ID	IDentity
KRR	Kernel Ridge Regression
LDA	Linear Discriminant Analysis
LDS	Low Density Separation
MAE	Mean Absolute Error
ML	Machine Learning
NPP	Nearest Point Problem
OCR	Optical Character Recognition
PCA	Principal Component Analysis
RBF	Radial Basis Function
QP	Quadratic Programming
RKHS	Reproducing Kernel Hilbert Space
RLS	Regularized Least Squares
RMSE	Root Mean Square Error
SDP	Semi-Definite Programming
SIFT	Scale Invariant Feature Transform
SLT	Statistical Learning Theory
SOCP	Second Order Cone Programming
SRM	Structural Risk Minimization
SV	Support Vector
SVM	Support Vector Machine
SVR	Support Vector Regression
SSL	Semi-Supervised Learning
TVK	Tangent Vector Kernel
VRM	Vicinal Risk Minimization
VSV	Virtual Support Vectors
USPS	United States Postal Service

Bibliography

[1] Belkin, M. Problems of Learning on Manifolds. Ph.D. dissertation, University of Chicago, 2003.

[2] Bezdec, J. and Hathaway, R. Convergence of alternating optimization. Neural, Parallel Sci.Comput.,, 11, pp. 351-368, 2003.

[3] Bhattacharyya, C., Pannagadatta, K. S., Smola, A. A Second Order Cone Programming Formulation for Classifying Missing Data. Proc. of Neural Inf. Proc. Systems., MIT press, Cambridge, 2004.

[4] Bi, J. and Zhang, T. Support Vector Classification with Input Data Uncertainty. Proc. of Neural Inf. Proc. Systems., MIT press, Cambridge, 2004.

[5] C. Bishop. Neural Networks for Pattern Recognition. Clarendon Press, Oxford, 1995.

[6] B.E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. Proc. of 5th ACM workshop on Computational Learning Theory, pp. 144-152, Pittsburgh, PA, 1992.

[7] Burges, C. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, Vol. 2, Number 2, p. 121-167, Kluwer Academic Publishers, 1998.

[8] C.J.C. Burges. Geometry and invariance in kernel-based methods. In B.Scholkopf, C.J.C. Burges, and A.J. Smola (eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1999.

[9] Chapelle, O., V. Vapnik and J. Weston Transductive Inference for Estimating Values of Functions. Advances in Neural Information Processing Systems 12, 1999.

[10] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, 2001. Vicinal Risk Minimization. In: T.K. Leen, T.G. Dietterich, and V. Tresp, (eds.), *Advances in Neural Information Processing Systems*, vol. 13, pp. 416-422.

[11] O. Chapelle and B. Scholkopf. Incorporating invariances in nonlinear SVMs. In T.G. Dietterich, S. Becker and Z. Ghahramani, (eds.),*Advances in Neural Information Processing Systems*, vol. 14, pp. 609-616. MIT Press, Cambridge, MA, USA, 2002.

[12] Chapelle, O., V. Vapnik, O. Bousquet and S. Mukherjee Choosing Multiple Parameters for Support Vector Machines. Machine Learning 46(1), 131-159, 2002.

[13] Chapelle, O., Zien, A. Semi-supervised Classification by Low Density Separation. In Proc. of AI&Statistics, 2005.

[14] Chapelle, O., Schölkopf, B., and Zien, A. (eds.) Semi-Supervised Learning. MIT Press, Cambridge, MA, in press, 2006.

[15] Cherkassky, V., Mulier, F. Learning From Data - Concepts, Theory, and Methods. John Wiley & Sons, USA, 1998.

[16] R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. Journal of Machine Learning Research, 1:143–160, 2001.

[17] D. DeCoste, M.C. Burl. Distortion-invariant recognition via jittered queries. In *Computer Vision and Pattern Recognition, CVPR-2000*, June, 2000.

[18] Fung, G., Mangasarian, O.L., and Shavlik, J. Knowledge-based support vector machines classifiers. Advances in Neural Information Processing Systems, vol. 15, Cambridge, MA, MIT Press, 2002.

[19] Evgeniou, T., Pontil, M., Poggio, T. Regularization networks and support vector machines. Advances in Computational Mathematics 13(1): 1-50, 2000.

[20] F. Girosi, and N. Chan, 1995. Prior Knowledge and the Creation of Virtual Examples for RBF Networks. Neural Networks Signal Processing Proceedings of the 1995/IEEE-SP/Workshop, IEEE Signal Processing Society, Cambridge, MA, 201-210, September 1995.

[21] Graepel, T., and Herbrich., R. Invariant Pattern Recognition by semidefinite programming machines. *Advances in Neural Information Processing Systems*, vol. 16, Cambridge, MA, MIT Press, 2003.

[22] I. Guyon (ed.), SVMs Application List. http://www.clopinet.com/isabelle/Projects/SVM/applist.html

[23] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using Suport Vector Machines. Machine Learning, 46, pp. 389-422, 2002.

[24] Haasdonk, B. Transformation Knowledge in Pattern Analysis with Kernel Methods. PhD thesis, Computer Science Department, University of Freiburg, May 2005.

[25] Haasdonk, B., Keysers, D. Tangent Distance Kernels for Support Vector Machines. In Proc. of International Conference on Pattern Recognition, Canada, Vol.2, pp. 864-868., 2002.

[26] Haasdonk, B., Vossen, A. and Burkhardt, H., Invariance in Kernel Methods by Haar-Integration Kernels. SCIA 2005, Scandinavian Conference on Image Analysis, pp. 841-851, Springer-Verlag, 2005.

[27] Hastie T., Tibshirani R. and Friedman J. The Elements of Statistical Learning. Springer press, 2001.

[28] Franz, M.O., Y. Kwon, C. E. Rasmussen and B. Scholkopf Semi-supervised kernel regression using whitened function classes. Proceedings of the 26th DAGM Symposium LNCS 3175, 18-26. (Eds.) Rasmussen, C. E., H. H. Bulthoff, M. A. Giese and B. Scholkopf, Springer, Berlin, Germany (2004)

[29] T. Jaakkola, and D. Haussler. Exploiting generative models in discriminative classifiers. In M.S.Kearns, S.A.Solla, D.A.Cohn (eds.) *Advances in Neural Information Processing Systems*, vol. 11, pp. 487-493, MIT Press, 1999.

[30] T. Joachims. Text categorzation with Support Vector Machines: learning with many relevant features. Proceedings of ECML'98 pp.137-142, Berlin, Springer, 1998.

[31] T. Joachims. Transductive Learning via Specral Graph Partitioning In Proc. of Int. Conference on Machine Learning, 2003.

[32] M. Kanevski, A. Pozdnoukhov, S. Canu, M. Maignan. Advanced Spatial Data Analysis and Modelling with Support Vector Machines. International Journal of Fuzzy Systems, Vol. 4, No. 1, pp. 606-616, 2002.

[33] M. Kanevski, A. Pozdnoukhov, S. Canu, M. Maignan, P. Wong, S. Shibli. Support Vector Machines for Classification and Mapping of Reservoir Data. A chapter from "Soft computing for reservoir characteri-zation and modelling", Springer-Verlag, pp. 531-558, 2001.

[34] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, 2000. A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. IEEE Transactions on Neural Networks, 11(1), pp.124−136.

[35] R. Kondor, T. Jebara, 2003. A Kernel Between Sets of Vectors. In proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC.

[36] Kondor, R., Jebara, T., Howard, A., (2004). Probability Product Kernels Journal of Vachine Learning Research, 5(2004), pp. 819-844.

[37] Leen, T. K. From data distributions to regularization in invariant learning. Neural Computation, vol. 7, no. 5, pp. 974-981, 1995.

[38] D.G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the Seventh International Conference on Computer Vision (ICCV'99), pages 1150-1157, Greece, 1999.

[39] J. Mariéthoz and S. Bengio. A comparative study of adaptation methods for speaker verification. In *International Conference on Spoken Language Processing ICSLP*, pages 581–584, Denver, CO, USA, 2002.

[40] Micchelli C. and Pontil, M. Learning the Kernel Function via Regularization. Journal of Machine Learning Research 6 (2005), 1099-1125.

[41] S. Mika, G. Raetsch, J. Weston, B. Scholkopf, K.R.Mueller. Fisher discriminant analysis with kernels. Proc. of Neural networks for signal processing IX, pp. 41-48, IEEE, 1999.

[42] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In European Conference on Computer Vision, pages 128-142. Springer, Copenhagen, 2002.

[43] P.J. Moreno, P. Purdy. A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications. In proc. of Advances in Neural Processing Systems, Vol. 16, 2003.

[44] K.R. Mueller, A.J. Smola, G. Raetsch, B. Scholkopf, J. Kohlmorgen, V.N. Vapnik. Predicting time series with support vector machine. Proceedings of ICANN'97 pp. 999-1004, 1997.

[45] Nesterov, Y., Nemirovskii, A., (1993) Interior Point Algorithms in Convex Programming. *Studies in Applied Mathematics, 13*, SIAM, Philadelphia.

[46] P. Niyogi, T. Poggio, and F. Girosi. Incorporating Prior Information in Machine Learning by Creating Virtual Examples. IEEE Proceedings on Intelligent Signal Processing, Vol. 86, No 11, 2196-2209, 1998.

[47] Fast Training of Support Vector Machines Using Sequential Minimal Optimization. Advances in Kernel Methods - Support Vector Learning, pp.185-208, MIT Press, Cambridge, MA, 1999.

[48] Pontil, M. and Verri, A. Support Vector Machines for 3D Object Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 6, pp. 637-646, June 1998.

[49] A. Pozdnoukhov. The analysis of kernel ridge regression algorithm. IDIAP RR-02-54, 2002.

[50] Pozdnoukhov A., Bengio, S., (2004). Tangent Vector Kernels for Invariant Image Classification with SVMs. *Proc. of Int. Conf. on Pattern Recognition*, Cambridge, UK.

[51] Pozdnoukhov A., Bengio, S., (2006). Graph-based Transformation Manifolds for Invariant Pattern Recognition with Kernel Methods. *Proc. of Int. Conf. on Pattern Recognition*, Hong Kong, 2006.

[52] Pozdnoukhov A., Bengio, S., (2006). Semi-Supervised Kernel Methods for Regression Estimation. *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing*, Toulouse, France, 2006.

[53] Pozdnoukhov A., Bengio, S., (2006). Improving Kernel Classifiers for Object Categorization Problems. *Proc. of ICML'05 workshop on Learning with Partly Classified Training Data*, Bohn, Germany, 2005.

[54] Pozdnoukhov A., Bengio, S., (2005). A Kernel Classifier for Distributions. IDIAP Research Report, IDIAP RR-05-32, 2005.

[55] Pozdnoukhov A., Bengio, S., (2006). From Samples to Objects: Invariances in Kernel Methods. In Pattern Recognition Letters Journal, Volume 27, Issue 10, pp. 1087-1097, 2006.

[56] Pozdnoukhov A., Kanevski, M., (2006). Monitoring Network Optimisation for Spatial Data Classification Using Support Vector Machines. International Journal of Environment and Pollution. Vol.28. 20 pp., 2006.

[57] Rabiner, L.R., (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, vol. 77, 2, February 1989, pp. 257-286.

[58] Rasmussen C.E., Williams C. Gaussian Processes for Machine Learning MIT press, 2006.

[59] C. Saunders, A. Gammerman, and V. Vovk. Ridge Regression in dual variables. Technical Report, Royal Holloway University of London, 1998.

[60] B. Scholkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J. C. Vorbrĺuggen, and B. Sendhoff, (eds.), Artificial Neural Networks ICANN'96, pp. 47-52, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

[61] B. Scholkopf, A.J. Smola, and K.R. Mueller. Kernel Principal Component Analysis. In Advances in Kernel Methods - Support Vector Learning, pp. 327-352, MIT press, Cambridge, MA, 1999.

[62] Scholkopf, B., Herbrich, R., and Smola, A., (2001) A Generalized Representer Theorem. In: Helmbold, D. and Williamson, B., (eds.) *Proc. of COLT/EuroCOLT 2001*, *LNAI2111*, pp. 416-426, Springer-Verlag, Berlin.

[63] Scholkopf, B., Smola, A.J. Learning with Kernels. MIT press, Cambridge, MA, 2002.

[64] Schneiderman, H., Kanade, T., (2000) A Statistical Method for 3D Object Detection Applied to Faces and Cars. *In the proc. of CVPR-2000*, pp.746-751.

[65] H. Schneiderman and T. Kanade. Object Detection Using the Statistics of Parts. International Journal of Computer Vision, Volume 56 , Issue 3, pp. 151-177, 2004.

[66] Shawe-Taylor J., Christianini N. Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.

[67] Simard, P., LeCun, Y., Denker, J., Victorri B., (1998). Transformation invariance in pattern recognition, tangent distance and tangent propagation. In: G. Orr and K. Muller, (eds.), *Neural Networks: Tricks of the trade.* Springer.

[68] Sindhwani, V., Niyogi, P., Belkin, M. Beyond the Point Cloud: from Transductive to Semi-supervised Learning In Proc. of ICML'05, Bonn, Germany.

[69] Smola, A.J., Schölkopf, B., Müller, K.-R. The connection between regularization operators and support vector kernels. Neural Networks, Volume 11, pp. 637-649, 1998.

[70] Smola, A.J., Scholkopf, B. A Tutorial on Support Vector Regression. Statistics and Computing, 1998. Invited paper.

[71] Tikhonov, A. N., Arsenin V. Y. Sollution of Ill-posed Problems W.H.Winston, Washington D.C., 1977.

[72] Tsochantaridis, I., Hofmann, T., Joachims, T., Yasemin, A., (2004). Support Vector Machine Learning for Interdependent and Structured Output Spaces. *21th Int. Conf. on Machine Learning,* Banff, Canada.

[73] R. Vanderbei LOQO: An Interior Point Code For Quadratic Programming. Technical Report SOR 94-15, Princeton University, 1994.

[74] V. Vapnik, 1998. Statistical Learning Theory. J.Wiley, NY, 1998.

[75] V. Vapnik, 2000. The Nature of Statistical Learning Theory. Second edition, Springer-Verlag, NY.