Learning Non-Linear Kernel Combinations Subject to General Regularization: Theory and Applications

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science

by

B. Rakesh Babu
200402007
rakeshbabu@research.iiit.ac.in

International Institute of Information Technology
Hyderabad, INDIA
June 2010

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Learning Non-Linear Kernel Combinations Subject to General Regularization: Theory and Applications" by Mr. B.Rakesh Babu, has been carried out under our supervision and is not submitted elsewhere for a degree.

Date	Prof. C. V. Jawahar, Professor, IIIT, Hyderabad	Dr. Manik Varma Researcher, Microsoft Research, Bangalore

To CVIT, IIIT Hyderabad the place which taught me ,
what an image is and what information can be derived from it.

To my Family.

Acknowledgements

I have been working on this thesis work for past three years. In this course I met many personalities who supported my research and I would like to acknowledge them. I would like to thank my advisors Prof. C. V. Jawahar and Dr. Manik Varma who gave me the introduction to the fields of machine learning and computer vision. Their encouragement and guidance made this work possible. The knowledge and valuable suggestions given by them provided the foundation for the work presented in this thesis. I thank Prof. P. J. Narayanan for providing excellent research lab environment at CVIT, IIIT Hyderabad. I would also like to thank Prof. Jayanthi Sivaswamy and Dr. Anoop Namboodiri, for co-ordinating and supporting various lab activities.

I would like to acknowledge Dr. Teo de Campos for his support while I was working on the problem of Character Recognition in Natural Images. I thank our lab assistants Satya and Phani for their help in CVIT. I am especially grateful to Microsoft Research for funding me on various occasions and encouraging my research. Further, I would like to thank my friends, colleagues and fellow CVIT-ians, specially OA (Sreekanth), Chetan and Venkat with whom I had many technical and philosophical discussions. I wish to especially acknowledge my seniors Praveen and Suman Karthik for insightful discussions.

Finally, I thank the almighty, my parents, my relatives and all those from CVIT and others who had at some point or the other helped me with their invaluable suggestions and feedback, and my research center, Center for Visual Information Technology (CVIT) for funding my MS by research in IIIT Hyderabad.

August 3, 2010

Abstract

Kernel methods are among the important recent developments in the field of machine learning with applications in computer vision, speech recognition, bio-informatics, etc. This new class of algorithms combine the stability and efficiency of linear algorithms with the descriptive power of nonlinear features. Kernel methods allow data to be mapped (implicitly) to a different space, which is often very high dimensional compared to the input space, so that complex patterns in the data become simpler to detect and learn. Kernel function maps the data implicitly into a different space. Support Vector Machines (SVMs) are one of the kernel methods which is widely successful for classification task. The performance of algorithm depends on the choice of the kernel. Sometimes, finding the right kernel is a complicated task. To overcome this, *learning the kernel* is the new paradigm which is developed in the recent years. For this, the kernel is parameterized as a weighted linear combination of base kernels. The weights of the kernel are jointly optimized with the objective of the task.

Learning both the SVM parameters and the kernel parameters is a Multiple Kernel Learning (MKL) problem. Many formulations of MKL are presented in literature. However, all these methods restrict to linear combination of base kernels. In this thesis, we show how the existing optimization techniques of MKL formulations can be extended to learn non-linear kernel combinations subject to general regularization on the kernel parameters. Although, this leads to non-convex problem, the proposed method retains all the efficiency of existing large scale optimization algorithms. We name the new MKL formulation as Generalized Multiple Kernel Learning (GMKL). We highlight the advantages of GMKL by tackling problems like feature selection and learning discriminative parts for object categorization problem. Here, we show how the proposed formulation can lead to better results not only as compared to traditional MKL but also as compared to state-of-the-art wrapper and filter methods for feature selection. In the problem of learning discriminative parts for object categorization, our objective is to determine minimal sets of pixels and image regions required for the task. We use the Multiple kernel learning to select the most relevant pixels and regions for classification. We then show how the framework can be used to enhance our understanding of the object categorization problem at hand, determine the importance of context and highlight artifacts in the training data. We also tackle the problem of recognizing characters in images of natural scenes in MKL framework. Traditionally it is not be handled well by OCR techniques. We assess the performance of various features (using bag-of-visual-words representation) based on nearest neighbor and SVM classification. Besides this, we investigate the appropriate representation schemes for recognition using MKL.

In short, the contributions of this thesis are:

1. Proposing new MKL formulation that can learn non-linear kernel combinations subject to general regularization on the kernel parameters.
2. Exploring the utility of multiple kernel learning formulations for feature selection and to the problem of learning informative parts for object category recognition.
3. Recognition of character images taken in natural scenes using the state of the art object recognition schemes. And also exploring the appropriate representation schemes for recognition using MKL.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Pattern recognition, a branch of artificial intelligence, studies the operations and design of systems which recognizes the patterns in data. This includes subfields, like discriminant analysis, feature extraction, error estimation, cluster analysis, etc. Most of recognition methods are either about supervised learning or unsupervised learning with applications in wide range of areas. In general, recognition schemes consists of learning algorithm where it learns a model based upon a given data (training data) and uses this model to classify unseen data (testing data).

Learning algorithms based on kernels have found to be successful in variety of tasks. Classification algorithms such as support vector machines [15, 16], regression algorithms such as kernel ridge regression, support vector regressions, [17, 18], and general dimensionality reduction algorithms such as kernel principle component analysis (KPCA) [19], kernel linear discriminant analysis (KLDA) [20] are based on kernel methods. Many of these methods are widely used in various fields e.g. bioinformatics [21–25], computer vision [26–28], speech recognition [29–31], data mining [32, 33], information retrieval [34, 35], pattern recognition [36], etc. This shows that the kernel methods have established themselves as powerful tools. These algorithms work by embedding the data into a feature space, and then searching for linear relations among the embedded data points. They employ a so called kernel function which intuitively computes the similarity between two points in the feature space. This information is contained in the kernel matrix, a symmetric and positive semidefinite matrix that encodes the relative positions of all points. Convergence of the training algorithms is ensured as long as the kernel matrix is symmetric and positive semidefinite. Figure 1.1 gives the overview of the kernel method. More details about kernel methods is discussed in Chapter 2. Note that classical kernel-based methods listed above are based on a single kernel.

The performance of these learning algorithms depends on the data representation. The kernel actually defines the similarity between two samples $\mathbf{x}, \mathbf{y}$, while defining an appropriate regularization term for the learning problem. In some situations, more flexible models are required. Recent works, show that using multiple kernels instead of a single one can enhance the interpretability of the output of algorithm and seen improved performances. One of the convenient approach which is considered in such situations

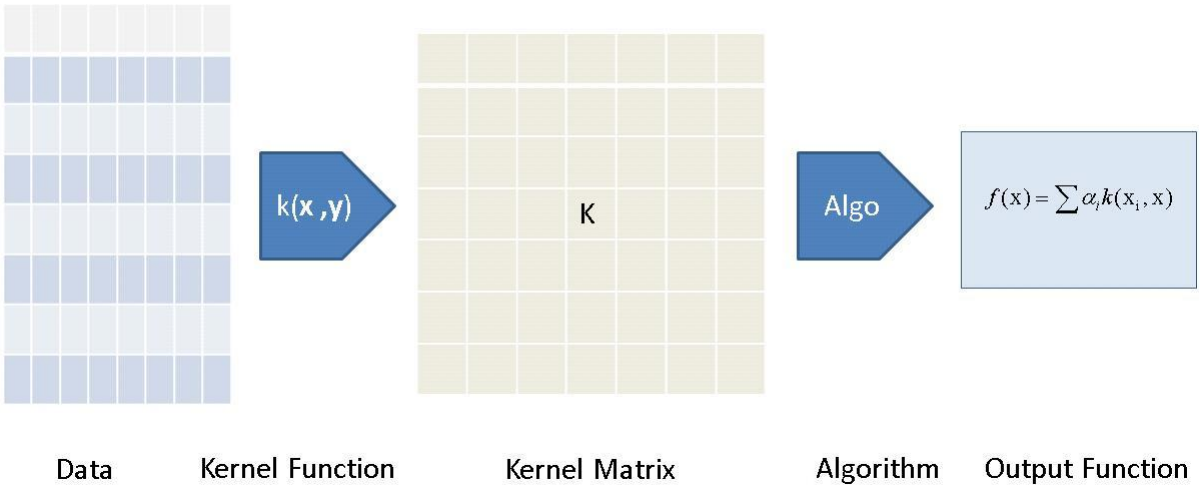


Figure 1.1: Overview of kernel method. Data with M samples is used to compute kernel matrix (M x M) using kernel function $k(\mathbf{x}, \mathbf{y})$. Then, learning algorithm uses this matrix to learn the model and computes output function.

is representing kernel function $k(\mathbf{x}, \mathbf{y})$ as conic combinations of base kernels.

$$k(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{K} d_l k_l(\mathbf{x}, \mathbf{y}) \text{ with } d_l \geq 0 \tag{1.1}$$

Each base kernel k_l may either use the full set of variables describing $\mathbf{x}$ or subsets of variables stemming from different data sources. Otherwise, the kernels k_l can simply be classical kernels like Gaussian kernels (i.e. $\exp^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$), etc with different parameters. Within this framework, the problem of data representation through the kernel is then transferred to the choice of weights d_l.

One of the most widely used kernel method is Support Vector Machines (SVMs) [15]. Support Vector Machines are basic tools in machine learning which are used for tasks such as classification, regression, *etc*. They find applications in diverse areas ranging from vision to bio-informatics to natural-language processing. The success of SVMs in these areas, is often dependent on the choice of a good kernel and features – which are typically hand-crafted and fixed in advance. However, hand-tuning kernel parameters can be difficult as selecting and combining appropriate sets of features. Learning both the SVM parameters and the weights (d_l) in a single optimization problem is a multiple kernel learning (MKL) problem. For binary classification, the MKL problem was first introduced by [2]. Intuition behind MKL formulation for classification is discussed in more detail in Section 2.6.

Multiple Kernel Learning (MKL) seeks to address the issue of appropriate data representation by learning the kernel from training data. In particular, it focuses on how the kernel can be learnt as a linear combination of given base kernels. Many MKL formulations have been proposed in the literature. In [4], it was shown that the MKL Block l_1 formulation of [3] could be expressed as a Semi-infinite Linear Program. Column generation methods and existing SVM solvers could then be used for efficient optimization and to tackle large scale problems involving as many as a million data points. Gradient

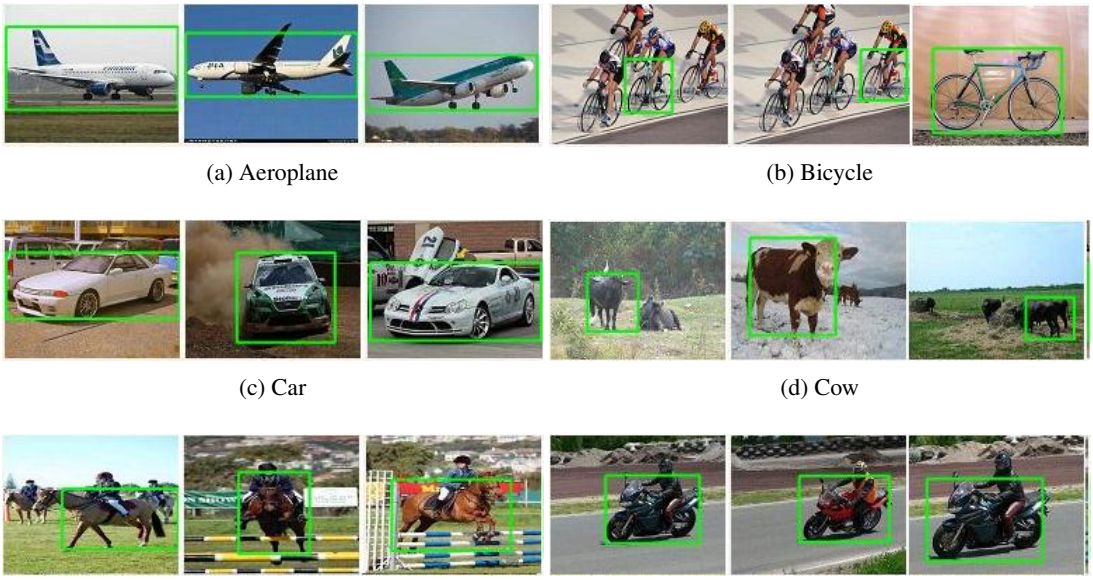
descent can be more efficient than solving a series of linear programs and [5] demonstrated that training time could be further reduced by nearly an order of magnitude on some standard machine learning datasets when the number of kernels is large. Simultaneously, MKL based algorithms have achieved very good results for bio-informatics [37, 38] and computer vision [6, 39] applications. These methods established the viability of MKL as a tool for tackling challenging real world problems.

1.2 Problem Statement

Learning the kernel is one of the popular paradigm developed for increasing performance and interpretability of the output of algorithm. MKL is a way of learning the kernel for the classification task. Many MKL formulations are presented in the literature [2–5]. But all the approaches are limited in that they focus on learning linear combinations of base kernels – corresponding to the concatenation of individual kernel feature spaces. Far richer representation can be achieved by combining kernels in other fashions. This raise the fundamental question of what could be other possible representations. This thesis addresses the problem of how the kernel can be learnt by using non-linear combinations. In specific, we address the following issues in this thesis i) Generalizing MKL to handle non-linear kernel combinations. In addition to kernel function, we also generalize the regularization on the kernel parameters. ii) How multiple kernel learning can be used for feature selection. We also demonstrate how the non-linear fashion of combining kernels boosts the performance when compared to linear manner of combining kernels. We demonstrate our results on standard machine learning and computer vision datasets. We also show how we can learn discriminative parts for object recognition in MKL framework. We demonstrate this on standard benchmark object recognition datasets. iii) Investigating and demonstrating the use MKL on the real world problem of character image recognition taken in natural scenes. Here, we use MKL for combining different features which captures different aspects like texture, edge, etc. The feature extraction methods we use here are widely used in object recognition literature.

The applications, on which we demonstrate the methods developed in the thesis, are of practical importance and have received wide attention in the field of machine learning and computer vision. Feature selection, the technique of selecting a subset of relevant features for building robust learning models is dealt with in this thesis. We also compare the proposed method with state-of-the-art methods proposed in literature. More details about the methods are presented in Chapter 5. Besides this, we deal with the problem of doing object categorization efficiently. For this we explore the use of multiple kernel learning to select the most relevant pixels and regions for classification. And thus do classification by using selected pixels or regions. We apply this for the problem of gender identification using minimal number of pixels. This has applications in video surveillance where efficient classifiers are needed. In Chapter 6, we deal with new problem of recognizing the character images taken in natural scenes. Solving this problem has practical applications in image retrieval, etc. The results obtained using the methods proposed in this thesis indicate learning non-linear (or generic) kernel combinations have greater impact in improving the performance in some cases up to 9%, and increase the understandability of certain problems.

The following section gives challenges involved in learning the kernel. Section 1.4 gives the details of various applications of MKL and section 1.5 presents the organization of the thesis.



(a) Aeroplane (b) Bicycle

(c) Car (d) Cow

(e) Horse (f) Motorbike

Figure 1.2: Results of object detection task using MKL based method proposed in [6] on some of categories from VOC Challenge dataset.

1.3 Challenges

Extracting useful knowledge from data is not always a trivial task. In the case of MKL, we aim at learning best generalized guaranteed classifier and simultaneously try to explore and find the best possible feature space. That is, we simultaneously find a separating hyperplane and the weights on each individual kernels. The weights are chosen so as to maximize the margin between the two classes. Trying to optimize both the objectives at a time is difficult as the two components are inter-linked to each other. In a way this can be framed as chicken-and-egg problem, where you do not know which one to start with, i.e., find the best classifier or best feature space. Besides this, when you have multiple heterogeneous data sources, it is extremely difficult to identify and explore the desired feature space. There is also necessity to find out which sources are needed to be given importance and which need not to be given. Extending to much more generic combinations and regularizations can make the formulations to be non-convex optimization problems. And, finding stable solutions to such types of problems are difficult.

Other challenges involved is that the optimization of such formulations is not straightforward. Many of these optimizations have the overhead of computational scalability, statistical stability which limits the applicability of solutions to small or medium scale problems. Scaling the solutions to larger problems is also one of the major challenges in solving the problem. Also scaling is essential in many practical situations.

1.4 Applications

Multiple kernel learning can be applied for wide range of applications in various fields. Fields include computer vision, machine learning, speech processing, bio-informatics, signal processing, data mining, etc. These applications use MKL either to increase interpretability of output or to increase prediction. Essentially these applications exploit one of the following views of multiple kernel learning.

- To Combine the multiple heterogeneous data sources.
- To obtain the optimal weights of different features used for the task.
- To interpret the sparsity after learning the weights of the kernels.

Designing and integrating kernels has proven to be an appealing approach to address several challenging real world applications. Specifically, problems involving multiple, heterogeneous data sources in computer vision, bio-informatics, audio processing problems, etc have been tackled successfully. For e.g. in computer vision it is used [6, 9, 39–41] to combine different features which capture various aspects like shape, color, texture, etc for image/object classification or object detection (see Figure 1.2). In the later chapters of this thesis we will investigate in greater details about their use for visual object classification in particular. Some applications in speech processing area are speaker verification [42] and speaker recognition. Some of the signal processing applications can be found in [43]. In the area of bio-informatics, it is used for various disease prediction and classification [37, 38, 44] tasks. There are also several other applications in machine learning to interpret the learning model [45, 46]. In this thesis, we address the problem of feature selection in detail.

1.5 Organization of the Thesis

Chapter 1 provides the broad overview of the thesis. The major contributions of the thesis are introduced. The challenges involved in tackling the problem and the possible applications where the solutions play a crucial role are discussed. Chapter 2 gives the background for reading the thesis. This gives detailed explanation of kernel methods and the multiple kernel learning. This chapter introduce the fundamental idea behind the kernel trick along with the elementary theory of kernel functions. Popular kernel method, Support Vector Machine is kernelized as an example to demonstrate the kernel trick. The dependency of the algorithm on choice of the kernel, popular kernels used in fields like computer vision and the fine details of MKL are given. Chapter 3 presents the literature survey on Multiple Kernel Learning. Here we review the development of MKL from initial work to current state-of-the-art methods.

In Chapter 4, we show how the MKL is generalized to learn non-linear kernel combinations subject to general regularization. This is achieved while retaining all the efficiency of existing large scale optimization algorithms. We name the new MKL formulation as generalized multiple kernel learning (GMKL). The theory and details of the formulation are given here. In Chapter 5 we demonstrate the applications of GMKL. Here we highlight the advantages of GMKL by tackling problems like feature selection and learning discriminative parts for object categorization problem. For feature selection, we

use various benchmark computer vision and machine learning datasets. Here we show how the proposed formulation can lead to better results not only as compared to traditional MKL but also compared to state-of-the-art wrapper and filter methods for feature selection. In the problem of learning discriminative parts for object categorization our objective is to determine minimal sets of pixels and image regions required for the task. We argue that information present in images can be redundant and, therefore, looking at the entire image might not be necessary for performing certain classification tasks. We use multiple kernel learning to select the most relevant pixels and regions for classification. We then show how the framework can be used to enhance our understanding of the object categorization problem at hand, determine the importance of context and highlight artifacts in the training data.

In Chapter 6, we tackle new problem of recognizing characters in images of natural scenes. In particular, we focus on recognizing characters in situations that would traditionally not be handled well by OCR techniques. We present results on an annotated database of images containing English and Kannada characters. The problem is addressed in an object categorization framework based on a bag-of-visual-words representation. We assess the performance of various features based on nearest neighbor and SVM classification. Besides this, we investigate the performance of MKL on the problem. Finally the conclusions of the thesis are given in Chapter 7.

Thus the contributions of thesis are : (i) Proposing new MKL formulation which is generalized to non-linear kernel combinations subject to general regularization on the kernel parameters (Chapter 4) . (ii) Exploring the utility of multiple kernel learning formulations for feature selection and to the problem of learning informative parts for object category recognition (Chapter 5). (iii) Recognition of perspectively imaged character images using the state of the art object recognition schemes. Also exploring the appropriate representation schemes for recognition using MKL (Chapter 6).

1.5.1 Note to the reader

Chapter 2 is written as a tutorial for the introduction to kernel methods and multiple kernel learning. It is not necessary to read this for understanding the thesis. However it is recommended for readers who are unfamiliar with kernel methods and have difficulty in understanding the multiple kernel learning problem. Readers who are familiar with the field may skip the chapter without losing continuity. Section 2.1 to Section 2.2 gives the introduction to kernel methods. In Section 2.3, we explain SVM for classification and how it can be kernelized. Section 2.4, 2.5 gives theory of kernels and some example kernel functions. And finally Section 2.6 gives the details of multiple kernel learning in detail. Readers who are familiar with kernel methods and not with multiple kernel learning can skip till Section 2.5 and can start reading from Section 2.6.

Chapter 2

Background on Kernel Methods and SVMs

2.1 Introduction to kernel methods

Over past decade kernel methods have received wide attention and have established themselves to be powerful tools in numerous domains. These methods are based on the similarities between the objects or samples they allow, e.g. the prediction of properties of new objects based on the properties of known ones (classification, regression) or identification of common subspaces or subgroups in otherwise unstructured data collections (dimensionality reduction, clustering).

In general, linear algorithms are widely used for many tasks such as dimensionality reduction, classification, because of its numerical and statistical stability. Linear relationships are easier to detect from data and most natural estimate of an unknown relationship among several variables. Principal Component Analysis [47], Linear Perceptron, Linear Predictive Coding [48] are some of the linear algorithms used for compression, modeling, prediction, etc. But, these methods are limited to only certain descriptive power. On the other hand non-linear algorithms have much more descriptive power than linear algorithms. These methods are extremely useful when tasks get complex as linear methods turns out to perform poor. But these non-linear algorithms are based upon non-linear functions which are difficult to estimate and has problems with stability (numerical and statistical) and convergence. In past, one of either methods is used depending upon situations as, there are no other class of methods which has descriptive power as well as numerical and statistical stability. Later on, kernel functions are introduced to draw the advantage of both the methods.

Kernel functions are first demonstrated in the introduction of Support Vector Machines (SVMs) [15] for the classification problem. These functions have successfully combined the advantages of both the linear algorithm and nonlinear functions. The method aims at building a linear classifier in a feature space that is nonlinearly related to the input space. This is done without explicitly accessing the feature space. The fundamental idea is that a complex relationship in the input data can be simplified by recoding the data in an appropriate manner. This paradigm is of little use for problems involving high-dimensional data. However, with the use of kernel function to indirectly access the recoded data via the

inner product makes estimation of non-linear functions feasible. Ever since the introduction of SVM, a number of successful linear algorithms such as PCA, LDA are kernelized [19, 20] using the kernel trick to incorporate the power of nonlinearity. The resulting algorithms are superior to their linear counterparts in terms of descriptive power, and are stable.

Any kernel function can be used with a kernelized algorithm without effecting the statistical properties, such as generalization capability of the algorithm (in case of classification algorithms). This allows domain specific knowledge (or prior) to be incorporated in to the kernel function without changing the algorithm. This modularity makes the development of powerful and stable algorithms feasible. Several other advantages of kernel methods will be described in the following sections. The underlying theory of kernel methods is covered in a number of books [49–51]. In the following section, different modules of kernel methods and basic methodology of each module is explained in detail. The use of kernel functions in SVM, how the performance of SVM can be improved by learning the kernel and then how these kernels are useful in computer vision are explained in later sections.

2.2 Overview of Kernel Methods

In general, kernel methods solution comprises of two parts. Firstly, a module that performs the mapping into the feature space and secondly, a learning algorithm designed to discover linear patterns in that space. The two main reasons why this approach is used are, (i) Detecting linear relations has been the focus of research in statistics and machine learning for decades, and the resulting algorithms are both well understood and efficient. (ii) There is a computational shortcut which makes it possible to represent linear patterns efficiently in high-dimensional spaces to ensure adequate representational power. This shortcut is called kernel trick with the help of kernel function.

The strategy adopted here is to embed the data into a space where the patterns can be discovered as linear relations. This is done in a modular fashion. Two steps with distinct components discussed earlier will perform this. The initial mapping component is defined implicitly by a so-called kernel function. This component will depend on the specific data type and domain knowledge concerning the patterns that are to be expected in the particular data source. The pattern analysis algorithm component is general purpose, and robust. Furthermore, it typically comes with a statistical analysis of its stability. The algorithm is also efficient, requiring an amount of computational resources that is polynomial in the size and number of data items even when the dimension of the embedding space grows exponentially.

In later section, we will introduce the main ingredients of kernel methods using SVM as example. Following four key aspects of the approach will be highlighted in the example.

1. Data items are embedded into a vector space called the feature space.
2. Linear relations are sought among the data points in the feature space.
3. The algorithms are implemented in such a way that the coordinates of the embedded points are not needed, only their pairwise inner products are required.
4. The pairwise inner products can be computed efficiently directly from the original data points using a kernel function.

These four observations will imply that, despite restricting ourselves to algorithms that optimize linear functions, approach will enable the development of a rich toolbox of efficient and well-founded methods for discovering nonlinear relations in the data. In the following section, linear version of SVM is first explained then kernel extension of it.

2.3 Support Vector Machines

Classification is a common task in machine learning. Given some data points with information of the class it belongs to, the goal of classification algorithm is learn the model to predict unseen samples which class it belongs to. Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. A Support Vector Machine is trained so that the direct decision function maximizes the generalization ability. Here, a data point is viewed as a $m-$dimensional vector (a list of m numbers), and we want to know whether we can separate such points with a $m-$dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One good choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. SVM chooses the hyperplane, so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines, is known as a *maximum margin classifier* .

2.3.1 Primal and Dual Formulation : Separable Case

Consider a two-class classification problem, let M $m-$dimensional training samples $\mathbf{x_i}(i = 1, \cdots, M)$ belong to either class 1 or class 2. And y_i be the corresponding labels which is 1 for class 1 and -1 for class 2. Consider problem to be separable and need to learn the decision function $\mathbf{w}^t\mathbf{x} + b$ where $\mathbf{w}$ is $m-$dimensional vector, b is a bias term. It is greater than zero for $y_i = 1$ and less than zero for $y_i = -1$. For controlled separability the following inequalities are used

$$\mathbf{w}^t\mathbf{x_i} + b \begin{cases} \geq 1 & \text{for } y_i = 1 \\ \leq -1 & \text{for } y_i = -1 \end{cases} \tag{2.1}$$

Above Equation can also be rewritten as and equivalent to

$$y_i(\mathbf{w}^t\mathbf{x_i} + b) \geq 1 \text{ for } i = 1, \cdots, M \tag{2.2}$$

In the hyperplane Equation

$$\mathbf{w}^t\mathbf{x_i} + b = c \tag{2.3}$$

when c = 0, it is the separating hyperplane which runs in the middle and parallel to the two hyperplanes with $c = 1$ and -1. The distance between these two hyperplanes is called the *margin*. Figure 2.1 shows the hyperplane formed when $c = 0, 1, -1$. It can been seen that there are many hyperplanes satisfying Equation (2.2). However, generalization ability of each the possible hyperplanes varies. Intuitively the hyperplane which has maximum margin will have more generalization ability and is called as the optimal separating hyperplane. Margin here is distance between the two hyperplanes $\mathbf{w}^t\mathbf{x_i} + b = 1$

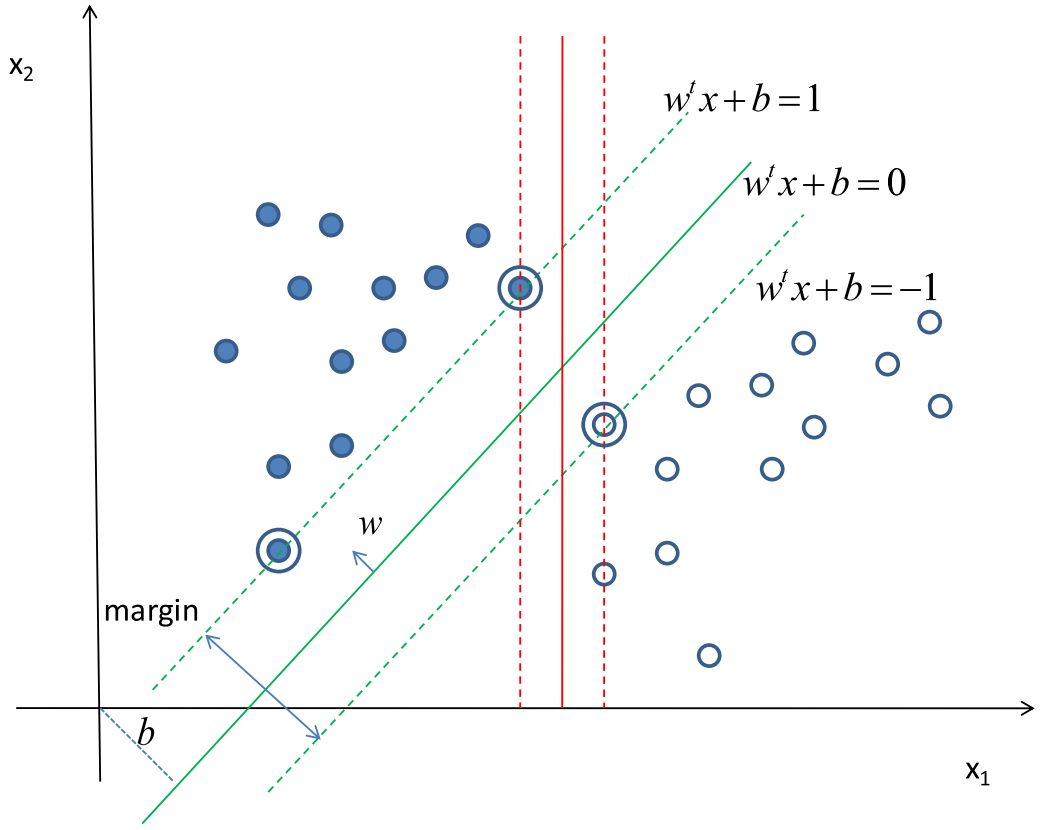


Figure 2.1: Maximum margin hyperplane in a two-dimensional data

and $\mathbf{w}^t\mathbf{x_i} + b = -1$ which is equivalent to $\frac{2}{\|w\|}$. The optimal separating hyperplane can be obtained by minimizing the following objective function J.

$$\begin{aligned} J = \quad & \frac{1}{2}\mathbf{w^t w} \\ \text{subject to} \quad & y_i(\mathbf{w}^t\mathbf{x_i} + b) \ \geq \ 1 \quad \forall\, i \\ y_i \in \ & \{-1, 1\} \end{aligned} \tag{2.4}$$

The square in the objective function of above formulation makes it a quadratic programming problem and *feasible solutions* exists as long as data is linearly separable. As the above formulation has quadratic objective functions, there exists non-unique solutions but the value of objective function is unique. This is one of the advantage of support vector machines over neural networks. The data which satisfy the equalities in first constraint of formulation given in Equation (2.4) are called *support vectors* . In Figure 2.1 the points which are rounded are support vectors.

By solving the formulation of Equation (2.4), $\mathbf{w}$ and b are estimated. So, the number of variables to be solved is the dimension of input vector plus one i.e $m + 1$. When the number of input variables are small above quadratic programming problem can be solved without much difficulty. But as discussed earlier we map the input space to a high-dimensional feature space, which might go infinite sometimes, finding the solution might not be feasible. For this, above formulation is converted into an equivalent dual where number of variables is equal to the number of training samples. This is converted into following unconstrained problem.

$$Q(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^t\mathbf{w} - \sum_{i=1}^{M} \alpha_i \left[y_i(\mathbf{w^T x_i} + b) - 1 \right] \tag{2.5}$$

where $\alpha_i \geq 0$ are the nonnegative Lagrangian multipliers. The optimal solution of Equation (2.5) is given by saddle point, which is minimized with respect to $\mathbf{w}$, b and maximized with respect to $\alpha_i (\geq 0)$, and it satisfies the following Karush-Kuhn-Tucker (KKT) conditions :

$$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0} \tag{2.6}$$

$$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial \mathbf{b}} = \mathbf{0} \tag{2.7}$$

$$\alpha_i\{y_i(\mathbf{w}^t\mathbf{x_i} + b) - 1\} = 0 \qquad \forall\, i \tag{2.8}$$

$$\alpha_i \geq 0 \qquad \forall\, i \tag{2.9}$$

Considering Equations (2.9), (2.8) together either $\alpha_i = 0$ or $\alpha_i \neq 0$ and $y_i(\mathbf{w}^t\mathbf{x_i} + b) = 1$ must be satisfied. The training samples for which $\alpha_i \neq 0$ are called *support vectors* . Using Equations (2.6), (2.7) and (2.9) we can deduce

$$\mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \mathbf{x}_i \tag{2.10}$$

and,

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \tag{2.11}$$

Using above two Equations and substituting in Equation (2.5), following dual problem is obtained. Maximize

$$\begin{aligned} Q_d(\alpha) = \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j \mathbf{x_i^t x_j} \\ \text{subject to } & \sum_{i=1}^{M} \alpha_i y_i = 0 \\ & \alpha_i \ \geq \ 0 \end{aligned} \tag{2.12}$$

Above problem is a concave quadratic programming problem and the solution exists as long as data is separable. The values of the primal J and dual objective $Q_d(\alpha)$ functions coincides at optimal solution and is called the zero duality gap. This is called hard margin support vector machines.

2.3.2 Primal and Dual Formulation : Non-Separable Case

When the data is inseparable, hard-margin support vector machines fails, as they cannot find a feasible solution. To handle inseparable case soft-margin support vector machines are proposed. Here the above formulation of a separable problem can be extended to a non separable one easily, by introducing a set of slack variables $\xi_i \ i = 1, \ldots, l$ in Equation (2.1) and becomes

$$\mathbf{w}^t \mathbf{x_i} + b \geq +1 - \xi_i \quad \text{for} \quad y_i = +1 \tag{2.13}$$

$$\mathbf{w}^t \mathbf{x_i} + b \leq -1 + \xi_i \quad \text{for} \quad y_i = -1 \tag{2.14}$$

$$\xi_i \geq 0 \quad \forall \, i \tag{2.15}$$

which can be rewritten as,

$$y_i(\mathbf{w}^t \mathbf{x_i} + b) \geq 1 - \xi_i \quad \text{for} \quad i = 1, \ldots, M \tag{2.16}$$

Slack variables ξ_i in the above Equation acts as penalty for misclassifying that particular sample. These variables are optimized by adding it to Equation (2.4) and the problem becomes minimizing,

$$\begin{aligned} J = \quad & \frac{1}{2} \mathbf{w}^t \mathbf{w} + C \sum_{i}^{M} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^t \mathbf{x_i} + b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i > 0, \ y_i \ \in \ \{-1, 1\} \quad \forall i \end{aligned} \tag{2.17}$$

Similar to the linearly separable case, this can be reformulated as minimizing,

$$Q(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{M}\xi_i - \sum_{i=1}^{M}\beta_i\xi_i - \sum_{i=1}^{M}\alpha_i\left[y_i(\mathbf{w^T x_i} + b) - 1 + \xi_i\right] \tag{2.18}$$

where $\alpha_i \geq 0$ and $\beta_i \geq 0$ are the nonnegative Lagrangian multipliers. Following KKT conditions are applied to get optimal solution.

$$\frac{\partial Q(\mathbf{w}, b, \xi, \alpha, \beta)}{\partial \mathbf{w}} = \mathbf{0} \tag{2.19}$$

$$\frac{\partial Q(\mathbf{w}, b, \xi, \alpha, \beta)}{\partial \mathbf{b}} = \mathbf{0} \tag{2.20}$$

$$\frac{\partial Q(\mathbf{w}, b, \xi, \alpha, \beta)}{\partial \xi} = \mathbf{0} \tag{2.21}$$

$$\alpha_i\{y_i(\mathbf{w}^t\mathbf{x_i} + b) - 1 + \xi_i\} = 0 \qquad \forall\, i \tag{2.22}$$

$$\beta_i\xi_i = 0 \qquad \forall\, i \tag{2.23}$$

$$\alpha_i \geq 0, \beta_i \geq 0, \xi_i \geq 0 \qquad \forall\, i \tag{2.24}$$

By substituting Equations (2.19), (2.20), (2.21) in (2.18) leads to following Equations.

$$\mathbf{w} = \sum_{i=1}^{M}\alpha_i y_i \mathbf{x}_i \tag{2.25}$$

$$\sum_{i=1}^{M}\alpha_i y_i = 0 \tag{2.26}$$

$$\alpha_i + \beta_i = C \quad \forall\, i \tag{2.27}$$

Substituting above three Equations in Equation (2.18) following dual problem is obtained. Maximize

$$\begin{aligned} Q_d(\alpha) = &\sum_{i}^{M}\alpha_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y_i y_j\langle \mathbf{x}_i, \mathbf{x}_j\rangle \\ \text{subject to} \quad &\sum_{i=1}^{M} y_i\alpha_i = 0, \quad \forall\, i \\ &C \geq \alpha_i \geq 0, \quad \forall\, i \end{aligned} \tag{2.28}$$

The only difference between the dual forms of soft-marign support vector machines and hard margin support vector machines is that α_i cannot exceed C. And decision function is given by

$$f(\mathbf{x}) = \sum_{i=1}^{M}\alpha_i y_i\langle \mathbf{x}_i, \mathbf{x}\rangle + b \tag{2.29}$$

2.3.3 Non - Linear SVM

Kernel Trick : The support vector machines discussed in the previous sections addresses the classification problem by building a linear classifier. The learnt classifier may not have high generalization ability though the hyperplanes are determined optimally. This can be solved by introducing nonlinearity through kernelizing the Support Vector algorithm. As discussed at the beginning of the chapter this can be done easily, by mapping the input space into a high-dimensional dot-product space called the *feature space*.

$$\mathbf{x} \in \Re^m \longmapsto \Phi(\mathbf{x}) \in F \subseteq \Re^p \tag{2.30}$$

By using the nonlinear vector function $\Phi(\mathbf{x}) = (\Phi_1(\mathbf{x}), \cdots, \Phi_p(\mathbf{x}))^t$ which maps the $m-$dimensional input vector $\mathbf{x}$ into the $p-$dimensional feature space. This feature space needs to be Hilbert space. The linear decision function in the feature space becomes,

$$\mathbf{w^t \Phi}(\mathbf{x}) + b \tag{2.31}$$

Now, $\mathbf{w}$ is an $p-$dimensional vector, as p increases the problem of solving formulation given in Equation (2.17) becomes difficult. On the other hand, in dual formulation, the algorithm needs the inner products between data points in the feature space F. It is worth taking the advantage of dual solution to solve the problem. The complexity of evaluating each inner product is proportional to the dimension of the feature space. The inner products can, however, sometimes be computed more efficiently as a direct function of the input features, without explicitly computing the mapping. In other words the feature-vector representation step can be by-passed. The class of functions which perform this direct computation are *kernel functions*.

Definition 1. *A* kernel *is a function k such that for all $\mathbf{x}, \mathbf{y} \in X$ satisfies*

$$k(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle,$$

where Φ is a mapping from X to an inner product feature space F

$$\Phi : \mathbf{x} \longmapsto \Phi(\mathbf{x}) \in F$$

The idea of kernel function can be illustrated with the help of following example. Consider the mapping of a two-dimensional input space $X \subseteq \Re^2$ with feature map,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \longmapsto \Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \in F = \Re^3$$

Here the feature map takes the data from two-dimensional to a three-dimensional space where the linear relations in feature space corresponds to quadratic relations in the input space. Now, the inner product in feature space can be evaluated as follows,

$$\begin{aligned}\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (y_1^2, y_2^2, \sqrt{2}y_1y_2) \rangle \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= (x_1y_1 + x_2y_2)^2 \\ &= \langle \mathbf{x}, \mathbf{y} \rangle^2\end{aligned}$$

Hence, the function

$$k(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$$

is the kernel function with F its corresponding feature space. Now, the inner product can be computed between the projections of two points into the feature space without explicitly evaluating their coordinates. Can every function be a kernel function and what set of kernel functions can be called as kernel function is discussed more widely in Section 2.4.

Kernel SVM : Now, kernelizing linear version of support vector machine is straightforward. In Equation (2.28) instead of accessing the input samples only via the inner product $\langle \mathbf{x_i}, \mathbf{x_j} \rangle$, can be made accessed through feature space through kernel function $k(\mathbf{x_i}, \mathbf{x_j})$. The dual problem in feature space is, maximize

$$\begin{aligned} Q_d(\alpha) &= \sum_{i}^{M} \alpha_i - \frac{1}{2} \sum_{i}^{M} \sum_{j}^{M} \alpha_i \alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) \\ \text{subject to} \quad & \sum_{i=1}^{M} y_i \alpha_i = 0, \quad \forall\ i \\ & C \geq \alpha_i \geq 0 \quad \forall\ i \end{aligned} \tag{2.32}$$

In general, $k(\mathbf{x_i}, \mathbf{x_j})$ is precomputed and stored in a matrix called kernel matrix ($\mathbf{K}$). For the inner product between $\mathbf{x_i}, \mathbf{x_j}$ is obtained by accessing $\mathbf{K}_{ij}$ instead of computing it on the fly. And the decision function is

$$f(\mathbf{x}) = \sum_{i=1}^{M} \alpha_i y_i k(\mathbf{x_i}, \mathbf{x}) + b \tag{2.33}$$

Since the discovery of kernelization, SVMs have been widely used for a number of applications involving classification and recognition. Note that till now we are solving two-class (binary) classification problem. This can be easily extended to multi-class classfication problems. Starting with naive approaches and more advanced formulations/approaches of multi-class classifications can be found in [52–56].

2.4 Valid Kernels

The kernel trick is to operate in feature space via a kernel function $k(,)$ The feature space is accessed indirectly via pairwise inner product. We now discuss the properties of these kernel functions here.

Definition 2. *Let $k(\mathbf{x}, \mathbf{y})$ be a real-valued symmetric function with $\mathbf{x}$ and $\mathbf{y}$ being m-dimensional vectors. For any set of data $\mathbf{x_1}, \cdots, \mathbf{x_M}$ and $\mathbf{a} = (a_1, \cdots, a_M)^t$ with M being any natural number, if*

$$\mathbf{a}^t \mathbf{K} \mathbf{a} \geq 0 \tag{2.34}$$

is satisfied (i.e., $\mathbf{K}$ is a positive semidefinite matrix), we call $k(\mathbf{x}, \mathbf{y})$ a positive semi-definite kernel, where

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x_1}, \mathbf{x_1}) & \dots & k(\mathbf{x_1}, \mathbf{x_M}) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ k(\mathbf{x_M}, \mathbf{x_1}) & \dots & k(\mathbf{x_M}, \mathbf{x_M}) \end{pmatrix}$$

Therefore, if k is a positive definite kernel then there exists a function $\Phi(\mathbf{x})$ that maps $\mathbf{x}$ into the dot-product feature space and $\mathbf{x}$ satisfies

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^t \Phi(\mathbf{y}) \tag{2.35}$$

The condition in Equation (2.34) is called *Mercer's condition* and kernel which satisfies this is called *Mercer kernel*, in general *kernel* . Now we see what are the general kernels functions used and how the new kernels can be designed from the existing set of kernels.

2.4.1 Kernels

Linear Kernels : In the linearly separable case, there is no need to map to high-dimensional space. In such cases we can use linear kernel,

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{x} \tag{2.36}$$

Polynomial Kernels : The polynomial kernel with degree d, where d is a natural number is given by,

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{x})^d \tag{2.37}$$

This is homogenous form of polynomial kernel, where as non-homogenous version of polynomial kernel is given by,

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{x} + 1)^d \tag{2.38}$$

When $d = 1$ it is linear kernel plus one. By adjusting b in the decision function both kernels produces same decision function. When $d = 2$ and $m = 2$ Equation (2.38) becomes

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2 \\ &= \Phi(\mathbf{x})^t \Phi(\mathbf{y}) \end{aligned}$$

where $\Phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2)^T$. Thus for d = 2 and m = 2 polynomial kernels satisfy Merecer's condition. This proof can also be extended easily for any value of d, m.

Radial Basis Function Kernels : The radial basis function (RBF) kernel is given by

$$k(\mathbf{x}, \mathbf{y}) = \exp^{(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)} \tag{2.39}$$

where γ is a positive parameter for controlling the radius. This is one of the most widely used kernel. The adjustable parameter γ plays a crucial role in the performance of the kernel, and should be tuned carefully depending upon the problem. If overestimated, the exponential will behave almost linearly and the higher-dimensional projection will starts to lose its non-linear power. On the other hand, if underestimated, the function will lack regularization and the decision boundary will be highly sensitive to noisy training data.

Hyperbolic Tangent (Sigmoid) Kernel: The Sigmoid Kernel comes from the Neural Networks field, where the sigmoid function is often used as activation function for artificial neurons.

$$k(\mathbf{x}, \mathbf{y}) = tanh(\mathbf{x}^t\mathbf{y} + c) \tag{2.40}$$

SVM model which uses a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

The above listed kernels are some of the standard kernels which are used widely. There many other kernels are presented in the literature which are specific to domain [57–67].

2.4.2 Kernel Design

Kernel function plays a key role in the performance of the kernel algorithms. New kernel function can be constructed from known kernel functions by performing certain operations. We now see the properties of positive semidefinite kernels that are useful for constructing new positive semidefinite kernels.

Result 1. *If*

$$k(\mathbf{x}, \mathbf{y}) = a, \tag{2.41}$$

where $a > 0,\ k(\mathbf{x}, \mathbf{y})$ *is positive semidefinite*

Proof. For any natural number M,

$$\mathbf{K} = (\sqrt{a}, \ldots, \sqrt{a})^t(\sqrt{a}, \ldots, \sqrt{a}) \tag{2.42}$$

$k(\mathbf{x}, \mathbf{y})$ is positive semidefinite. □

Result 2. *If* $k_1(\mathbf{x}, \mathbf{y})$ *and* $k_2(\mathbf{x}, \mathbf{y})$ *are positive semidefinite kernels,*

$$k(\mathbf{x}, \mathbf{y}) = a_1k_1(\mathbf{x}, \mathbf{y}) + a_2k_2(\mathbf{x}, \mathbf{y}) \tag{2.43}$$

is also positive semidefinite, where a_1 *and* a_2 *are positive.*

Proof. For any M, a_i and $\mathbf{x_i}$

$$\begin{aligned} \mathbf{a}^t\mathbf{K}\mathbf{a} &= \mathbf{a}^t(a_1\mathbf{K_1} + a_2\mathbf{K_2})\mathbf{a} \\ &= a_1\mathbf{a}^t\mathbf{K_1}\mathbf{a} + a_2\mathbf{a}^t\mathbf{K_2}\mathbf{a} \\ &\geq 0 \end{aligned} \tag{2.44}$$

Therefore, $k(\mathbf{x}, \mathbf{y})$ is positive semidefinite. □

Result 3. *If $k_1(\mathbf{x}, \mathbf{y})$ and $k_2(\mathbf{x}, \mathbf{y})$ are positive semidefinite kernels,*

$$k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y})k_2(\mathbf{x}, \mathbf{y}) \tag{2.45}$$

is also positive semidefinite.

Proof. To prove this it is sufficient to show that if $M \times M$ matrices $B = \{b_{ij}\}$ and $C = \{cij\}$ are positive semidefinite, $a_{ij}b_{ij}$ is also positive semidefinite. Since B is positive semidefinite, through mercer condition we can say that B is expressed by $B = F^tF$, where F is an $M \times M$ matrix. Then $b_{ij} = \mathbf{f_i}^t\mathbf{f}$, where $\mathbf{f_j}$ is the **j**th column vector of F. Thus for any arbitrary $(a_1, \ldots, a_M)$,

$$\begin{aligned} \mathbf{a}^t\mathbf{K}\mathbf{a} &= \sum_{i,j=1}^{M} h_i h_j a_{ij} b_{ij} \\ &= \sum_{i,j=1}^{M} h_i h_j \mathbf{f_i}^t \mathbf{f_j} b_{ij} \\ &= \sum_{i,j=1}^{M} (h_i \mathbf{f_i})^t (h_j \mathbf{f_j}) b_{ij} \\ &\geq 0 \end{aligned} \tag{2.46}$$

Thus $k(\mathbf{x}, \mathbf{y})$ is positive semidefinite. □

In the next section we see some of the popular kernels used in specific field computer vision.

2.5 Kernels for computer vision

Over the last years, *kernel methods* have established themselves as powerful tools for computer vision researchers as well as for practitioners. All the methods for regression, dimensionality reduction, outlier detection, clustering, recent methods of non-classical techniques for the prediction of structure data, for the estimation of statistical dependency, and for learning the kernel function itself are illustrated with successful examples applications in the recent computer vision research literature.

Images and videos are a data source with a very special characteristic: because each pixel represents a measurement. Images are typically very high dimensional. Smaller resolution image of 256 x 256 will contain more than 65k pixels and moving to higher resolution will be of even more high in dimensional. This is the main reason why kernels methods are widely applicable in the field of computer vision. Therefore, Computer Vision researchers have given special attention on finding good data representations and algorithms to tackle problems, such as (i) Optical character recognition: classify images of handwritten or printed letters or digits [68], (ii) Object classification: classify natural images according to the object category they contain [39], (iii) Action recognition: classify video sequences based on the action performed in them [69], (iv) Image segmentation: partition an image into the subregions that correspond to different image aspects, e.g. background or foreground [70], (v) Content Based Image retrieval: find images that are most similar to a query image from a collection or database [71]. (vi) Object Detection: identify the boundary of the object present in the image [6]

Figure 2.2: Sample images of class elk taken from Caltech 256 [7] dataset. Notice the variations in color, location, contrast in background. And there are also lots of variation in pose & structure of the object class.

Kernel methods have proved successful in all of these areas, mainly because of their interpretability and flexibility. By constructing a kernel function one can integrate knowledge that humans have about the current problem. And this leads to improved performance compared to pure black-box methods that do not allow the integration of prior knowledge. There is much research in designing promising kernels which is specific to the task. Once it is designed it can be re-used in any kernel method not only just in the context it was originally designed. This gives researchers as well as practitioners a large pool of established kernel functions to choose from, thereby increasing the chances of finding a well-performing one. In the following, we introduce some of the existing kernels, the assumptions they were based on, and their applicability to practical computer vision tasks starting with some basics.

2.5.1 Interest points, Descriptors and Bag-of-Words

In computer vision, visual descriptors or image descriptors are descriptions of the visual features of the contents in images or videos, which are calculated at certain points called *interest points*. At these points, descriptor describe elementary characteristics such as the shape, the color, the texture or the motion. These descriptors carry the knowledge of the objects and events found in a video, image and this is used to for further processing.

But advanced computer vision tasks require generalization not only between different views of the same objects, but also between many different objects that share a semantic aspect, e.g. animals of the same species. The visual variations within such a class can be very large, and is illustrated with an example in Figure 2.2. These variations generally occur due to change of pose, truncation or occlusion. But typical parts are often common for all object instances. Part-based representations of natural images have been developed to overcome all of these problems. They are based on the idea of treating the image as collections of many local parts instead of as single object with global properties.

To find relevant parts of the image, in general one applies a set of operators for the detection of interest points. These operators comprises of low-level differential filters based on differences of Gaussian or Wavelet coefficients , etc. It is shown in practice that interest points on a regular grid or random locations and scales [72] work well. Each region of interest defines a small image from which one calculates an invariant representation, often called a descriptor. The popular SIFT descriptor [72] does this by combining several ideas. Many other descriptors [73–77] have been developed that follow similar or some other design. Many of these descriptors are used for problems which are addressed in this thesis

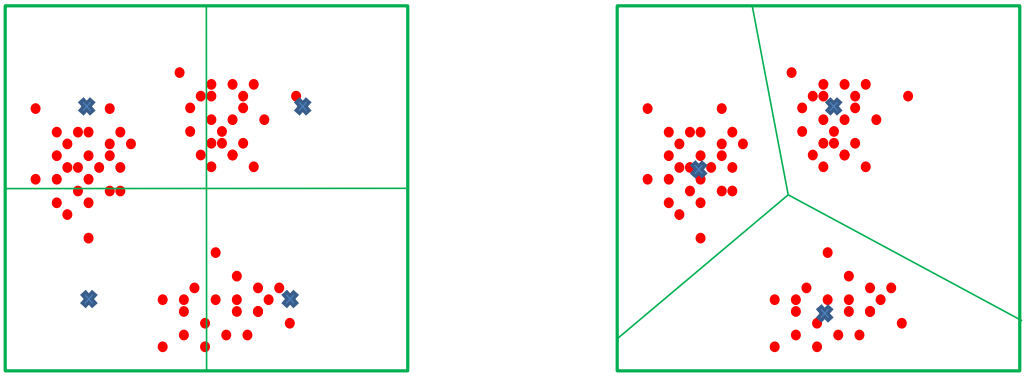


Figure 2.3: The descriptors which occur in natural images do not lie uniform in the space of all possible descriptors, but they form clusters. BOW (right) divides the descriptor space into Voronoy cells that respect the cluster structure but SPK(left) does not do this. Image courtsey [8]

and more details of them are given at appropriate place. After this first preprocessing step, the image is represented as a set of descriptor vectors, one per region of interest in the image. All descriptors vectors are of the same length, typically between 20 and 500 dimensions. The number of regions and descriptors varies depending on the image contents. Depending on the method for interest point detection and the resolution of image, the number of interest points per image vary.

After interest point detection, each image is abstracted by several local patches. Feature representation methods deal with representation of patches as numerical vectors. Natural images have inherent *regularities* that cause the extracted descriptors vectors to form clusters in the descriptors space. For example, edges and corners are typically much more frequent than, e.g., checker board-like patterns. On one hand, a large number of grid cells will stay empty, and on the other hand, existing clusters might be split apart.

The vector representing patches are represented in the next level using *codewords*. A codeword can be considered as a representative of several similar patches. One simple method is performing K-means clustering over all the vectors [78]. Codewords are then defined as the centers of the learnt clusters. The number of the clusters is the size of codebook. This is also called as *vocabulary size*. Thus, each patch in an image is mapped to a certain codeword through the clustering process. As a simplest representation, we count for each cluster center, how often it occurs as a nearest neighbor of a descriptor in $\mathbf{x}$ and form the resulting K-bin histogram. This construction is often called bag of visual words, since it is similar to the bag-of-words concept in natural language processing.

2.5.2 Pyramid Match Kernel

Pyramid match kernel [79] is a fast kernel function (satisfying Mercer's condition) which has been built over these descriptors and proven themselves in the tasks like object recognition. The complexity of

comparing two images in part-based representation can be made linear instead of quadratic by quantizing the space of possible descriptor values. The pyramid match kernel (PMK) [79] does so well by subdividing the d-dimensional space of image descriptors into a hierarchy of axis parallel cells in a data dependent way. In the finest layer, each descriptor lies in a cell of its own. Coarser layers are built by merging neighboring cells in any dimension. This construction is repeated until the coarsest layer has only one cell containing all descriptors. It is defined as

$$k_{PMK}(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{L} 2^l \sum_{j=1}^{2^{l-1}} \min(h^{l,j}(\mathbf{x}), h^{l,j}(\mathbf{y})) \tag{2.47}$$

where $h^{l,j}(\mathbf{x})$ are histograms of, how many features of $\mathbf{x}$ falling into $j-$th cell of $l-$th pyramid level. This kernel has been successfully demonstrated on caltech 101 [14] and ETHZ databases [80]. But quantization of the descriptor space by a regular grid, as used by the pyramid match kernel, does not reflect proper clustering, see Figure 2.3.

2.5.3 Kernels for BOW Representations

The representation of images as feature count histograms leaves us with many possibilities which kernel function to apply on them. A direct analogue of the pyramid match kernel Equation (2.47) is the histogram intersection kernel [67]:

$$k_{HI}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{V} min(h^i, h'^i) \tag{2.48}$$

where we write $h = (h^1, \ldots, h^V)$ for the $V-$bin histogram representation of $\mathbf{x}$ and analogously h' for the histogram of $\mathbf{y}$.

For fixed length histograms we can apply all kernels defined for vectors, e.g. linear, polynomial or Gaussian. If the number of feature points differs between images, it often makes sense to first normalize the histograms, e.g. by dividing each histogram bin by the total number of feature points. This allows the use of kernels for empirical probability distributions, e.g. the Bhattacharyya kernel

$$k_{bhattacharyya}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{V} \sqrt{h^i, h'^i} \tag{2.49}$$

Another popularly used kernel in Computer Vision is χ^2-kernel:

$$k_{\chi^2}(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma \sum_{i=1}^{V} \frac{(h^i - h'^i)^2}{h^i + h'^i}\right) \tag{2.50}$$

which has shown very good performance, in the tasks like object recognition [39], object detection [81].

2.5.4 Spatial Pyramid Kernel

The bag of visual words model completely ignores the spatial structure information from the image. However, in some tasks spatial information can be a valuable source of information, e.g. if one wants to recognize scene like highway where sky regions tend to occur much more frequently at the top of the image than at the bottom. Consequently, the idea of local histograms has proved useful in this setup as well. In the place of global visual word histogram, a number of local histograms are formed, typically in a pyramid structure from coarse to fine as similar to pyramid match kernel. Each sub-histogram has V bins and counts how many descriptors with center point in the corresponding pyramid cell have a specific codebook vector as nearest neighbor. Then, either all local histograms are concatenated into a single larger histogram , or separate kernel functions are applied for each level and cell, and the resulting kernel values combined into a single spatial pyramid score, e.g. by a weighted sum [82]:

$$k_{SP}(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{L} d_l \sum_{j=1}^{2^l-1} k(h_{(l,j)}, h'_{(l,j)}) \tag{2.51}$$

where L is the number of levels, d_l is a per-level weight factor and $h_{(l,k)}$, $h'_{(l,k)}$ are the local histograms of $\mathbf{x}$, $\mathbf{y}$ respectively. The base kernel k is typically chosen from the same selection of histogram kernels as above, with or without separate histogram normalization.

2.6 Learning the Kernel

Kernel method poses many advantages other than nonlinearity such as modularity, ability to work with heterogeneous description of data, etc. The major issue in the kernel methods is the choice of kernel function. The kernel function defines the geometry of space in which an algorithm operates and this is crucial for the performance of that algorithm in that space. In general kernel methods use a single fixed kernel function. Different kernel functions induce different feature space embeddings and are therefore differently well suited for a given problem. Finally, the choice of the kernel is task dependent. The quality of a kernel is determined by how well the trained kernel method performs in that particular task at hand, e.g. in the case of a classifier by the accuracy on unseen data points. Although, many estimators for the generalization error have been developed and used for parameter selection, e.g. *cross-validation* and *bootstrapping*, which work by iterating between training and test procedures on different parts of the training set.

2.6.1 Kernel Target Alignment

The idea of learning the kernel matrix has originated from [1]. which defines an alignment between a kernel and a set of labels. The intuition of kernel target alignments (KTA) [1] is that the values of a good kernel function k should resemble the values of a (hypothetical) ideal kernel l. This ideal kernel or target kernel is constructed by $l(\mathbf{x}, \mathbf{y}) = y_i y_j$ with $y_i \in \{-1, +1\}$. The alignment between kernel k, l is defined as

$$A(k,l) = \frac{\langle \mathbf{K}, \mathbf{L} \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \mathbf{L}, \mathbf{L} \rangle_F}} \tag{2.52}$$

where $\mathbf{K}, \mathbf{L}$ is the kernel matrix using kernel k, l in feature space F respectively. This can also be viewed as the cosine angle between two bi-dimensional vectors $\mathbf{K}$ and $\mathbf{L}$. Substituting $\mathbf{L} = \mathbf{y}\mathbf{y}^t$ where $\mathbf{y}$ is vector of labels of the training samples, then

$$\begin{aligned} A(k,l) &= \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^t \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \mathbf{y}\mathbf{y}^t, \mathbf{y}\mathbf{y}^t \rangle_F}} \\ &= \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^t \rangle_F}{m\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F}} \end{aligned} \tag{2.53}$$

To select one kernel function out of a set of alternatives, we choose the kernel function that maximizes $A(k,l)$. Since this procedure does not require to train and evaluate a classifier, it is in particular faster than, e.g., multiple cross-validation runs. Another advantage of the kernel alignment score is its differentiability with respect to the kernel function k. For kernels that depend smoothly on real-valued parameters, it therefore possible to find locally optimal parameters combination by gradient-descent optimization.

2.6.2 Multiple Kernel Learning

For many tasks the choice of representation and features depends on the applications. For instance in computer vision for a problem color, texture, or edge orientation might be the most relevant cue. Most often, one finds that different aspects are important at the same time, and one would like to find a kernel function that reflects the aspects of several kernels at the same time.

Kernel methods in general are well suitable for such feature combinations. Constructing Kernel functions, the sum and product of existing kernels are kernels again, equally reflecting the properties of all base kernel. However, in situations, where we believe that some kernels are more important than others, we might prefer a weighted linear combination of kernel instead of their unweighted sum. *Multiple kernel learning* (MKL) allows us to find the weights of such linear combinations. The intuition here is that kernel or combination of kernels gives rise to a margin when used in the training of a support vector machine, and due to the linear kernel construction, we can find an explicit expression for the size of the margin. The concept of maximum margin learning tells us to prefer classifiers with a large margin between the classes. MKL procedure jointly finds the SVM weight vector and the linear combination weights of the kernel functions that realized the generalized linear classifier of maximal margin. See Figure 2.4 for an illustration.

Linear kernel combinations : Let $k_1, \cdots, k_K$ be kernel functions, $k_i : \chi \times \chi$ with induced Hilbert spaces $\mathcal{H}_i$ and feature maps Φ_i. Now the interest lies in finding the best SVM classifier for kernel

$$k(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{K} d_l k_l(\mathbf{x}, \mathbf{y}) \tag{2.54}$$

with $d_i \geq 0$.

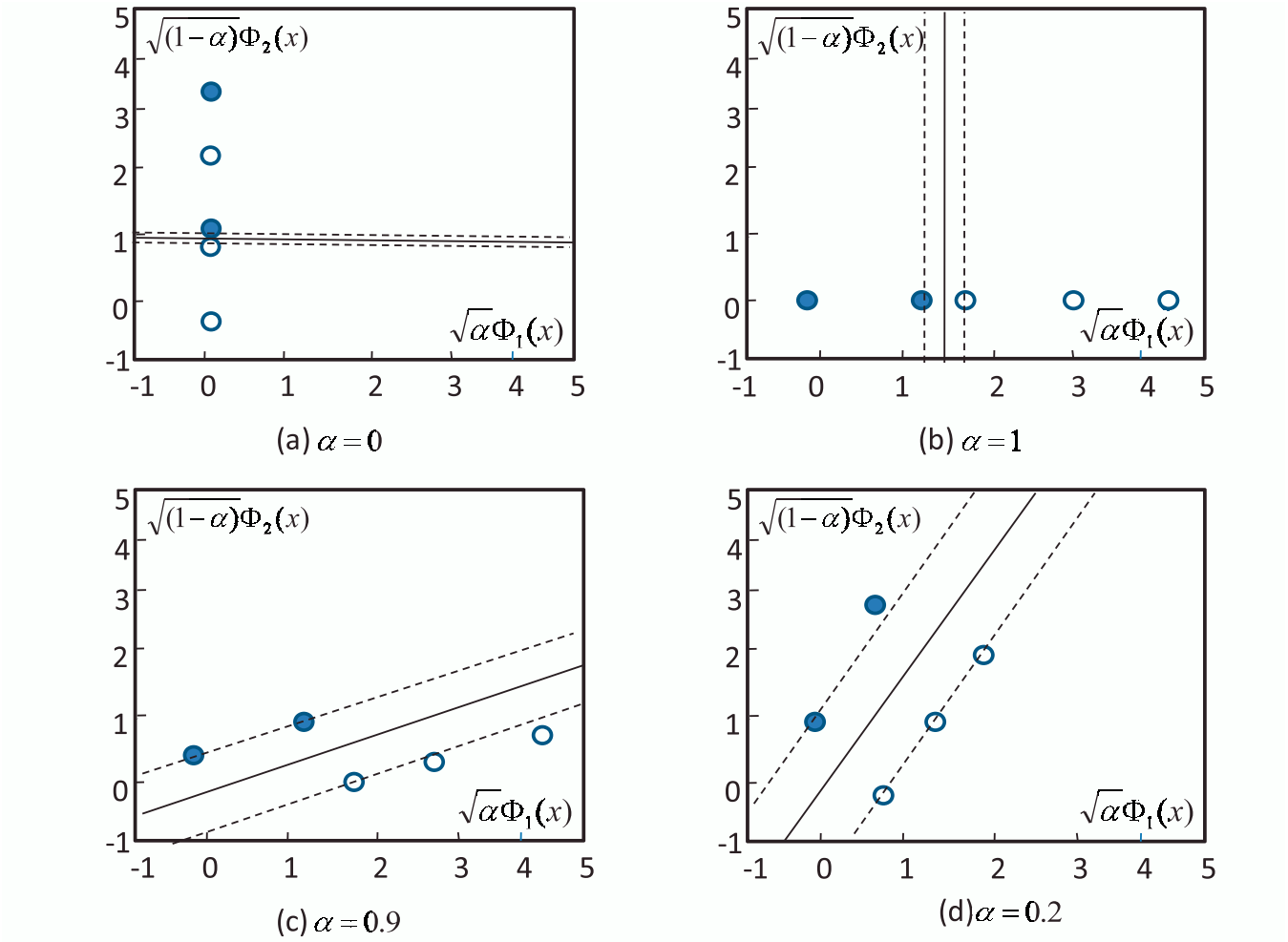


Figure 2.4: Given two kernels k_1, k_2 with feature maps Φ_1, Φ_2 then consider the kernel formed through linear combination $k = \alpha k_1 + (1-\alpha)k_2$ with induced feature space $(\sqrt{\alpha}\Phi_1, \sqrt{1-\alpha}\Phi_2)$. Plots corresponding to $\alpha = 0, 1, 0.9, 0.2$ can be found in (a),(b),(c),(d) respectively. It is clear that data is not much separable in the original features space (a), (b) when compared to to feature spaces (c),(d). Image courtsey [8]

If $d_1, \ldots, d_K$ are fixed $\Phi(\mathbf{x}) = (\sqrt{d_1}\Phi_1(\mathbf{x}), \ldots, \sqrt{d_K}\Phi_K(\mathbf{x}))^t$ since this constructs same scalar product as k:

$$\begin{aligned}\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle &= \sum_{l=1}^{K} d_l \langle \Phi_l(\mathbf{x}), \Phi_l(\mathbf{y}) \rangle \\ &= \sum_{l=1}^{K} d_l k_l(\mathbf{x}, \mathbf{y}) \\ &= k(\mathbf{x}, \mathbf{y})\end{aligned} \tag{2.55}$$

Finding the best coefficients enables better construction of feature space. Therefore, to find the best coefficients for the linear combination kernel $k_{opt} = \sum_l d_l k_l$, following objective function is presented in [39]. Minimize

$$\begin{aligned} J =& \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i}^{M}\xi_i + \sum_{l}^{K} d_l\sigma_l \\ \text{subject to} \quad & y_i(\mathbf{w}^t\Phi(\mathbf{x_i}) + b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i > 0,\ y_i \in \{-1, 1\} \quad \forall i \\ & \mathbf{d} \geq 0, \mathbf{Ad} \geq \mathbf{p} \\ & \Phi(\mathbf{x_i})^t\Phi(\mathbf{x_j}) = \sum_{l=1}^{K} d_l \Phi(\mathbf{x_i})_l^t \Phi(\mathbf{x_j})_l \end{aligned} \tag{2.56}$$

where $\mathbf{d}$ are kernel parameters and $\mathbf{A}, \mathbf{p}$ are the parameters to include prior knowledge on kernel parameter $\mathbf{d}$. The objective function in formulation (2.56) is similar to l_1 soft margin SVM formulation (2.17). Given the misclassification penalty C, it maximizes the margin while minimizing the hinge loss on the training set. The only addition to it is kernel parameter also optimized along with SVM parameters. In general most of the weights will be zero depending on the parameters σ which encode prior preferences for particular kernels. The l_1 regularization thus prevents over-fitting as only few kernels are being used at the end. And there are two additional constraints which are added in comparison with standard SVM. The first, $\mathbf{d} \geq 0$, this is to ensure that weights are interpretable and also leads to a much more efficient optimization problem. The second, $\mathbf{Ad} \geq \mathbf{p}$ it to encode prior knowledge about the kernel parameters. The final condition is just restatement of Equation (2.54).

Similar to the Equation (2.17), above objective function can be reformulated as maximizing

$$\begin{aligned} Q(\mathbf{w}, b, \xi, \mathbf{d}, \alpha, \beta, \gamma, \delta) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{M}\xi_i + \sum_{l=1}^{K} d_l\sigma_l - \sum_{i=1}^{M}\alpha_i\left[y_i(\mathbf{w^T\Phi}(\mathbf{x_i}) + b) - 1 + \xi_i\right] \\ - \sum_{i=1}^{M}\beta_i\xi_i - \sum_{l=1}^{K}\gamma_l d_l - \sum_{l=1}^{K} d_l\boldsymbol{\delta}^t\mathbf{A_l} + \mathbf{p^t}\boldsymbol{\delta} \end{aligned} \tag{2.57}$$

1

[1]Note that $\boldsymbol{\delta^t}(\mathbf{Ad} - \mathbf{p})$ is rewritten as $\sum_{l=1}^{K} d_l\boldsymbol{\delta}^t\mathbf{A_l} - \mathbf{p^t}\boldsymbol{\delta}$

where $\alpha_i, \beta_i, \gamma_l, \delta_l$ are the nonnegative Lagrangian multipliers and $\mathbf{A_l}$ is lth column of the matrix $\mathbf{A}$. On differentiation with respect to $\mathbf{w}, b, \xi_i, d_k$, we have,

$$\frac{\partial Q}{\partial \mathbf{w}} = \mathbf{0} \Longrightarrow \mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \Phi(\mathbf{x_i}) \tag{2.58}$$

$$\frac{\partial Q}{\partial b} = \mathbf{0} \Longrightarrow \sum_{i=1}^{M} \alpha_i y_i = 0 \tag{2.59}$$

$$\frac{\partial Q}{\partial \xi_i} = \mathbf{0} \Longrightarrow C = \alpha_i + \beta_i \tag{2.60}$$

$$\frac{\partial Q}{\partial d_l} = \mathbf{0} \Longrightarrow \sigma_l = \gamma_l + \boldsymbol{\delta}^t \mathbf{A_l} + \sum_{i=1}^{M} \alpha_i y_i (\mathbf{w}^t [\mathbf{0_1} \cdots \mathbf{\Phi_l}(\mathbf{x_i}) \cdots \mathbf{0_K}]^\mathbf{t}) \frac{\mathbf{1}}{\mathbf{2\sqrt{d_l}}} \tag{2.61}$$

$$\alpha_i \{ y_i (\mathbf{w}^t \mathbf{x_i} + b) - 1 + \xi_i \} = 0 \quad \forall\, i \tag{2.62}$$

$$\beta_i \xi_i = 0 \quad \forall\, i \tag{2.63}$$

$$\alpha_i \geq 0, \beta_i \geq 0, \xi_i \geq 0, d_l \geq 0 \quad \forall\, i, l \tag{2.64}$$

$$\gamma_l d_l = 0 \quad \forall\, l \tag{2.65}$$

$$\boldsymbol{\delta^t} (\mathbf{Ad} - \mathbf{p}) = \mathbf{0} \tag{2.66}$$

where $\mathbf{0}_l$ is a vector containing all zeros of size $\Phi_l(\mathbf{x_i})$. Equations from (2.62) to (2.66) are KKT conditions and substituting Equations from (2.58) to (2.61) in (2.57) we get following dual problem (for more detailed derivation see Appendix A). Maximize

$$\begin{aligned} Q_d &= \mathbf{1^t} \boldsymbol{\alpha} + \mathbf{p^t} \boldsymbol{\delta} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad 0 \leq \boldsymbol{\delta}, \quad \mathbf{1}^t \mathbf{Y} \boldsymbol{\alpha} = 0 \\ & \frac{1}{2} \boldsymbol{\alpha^t} \mathbf{Y K_l Y} \boldsymbol{\alpha} \leq \sigma_l - \boldsymbol{\delta^t} \mathbf{A_l} \end{aligned} \tag{2.67}$$

where the non-zero α_i correspond to the support vectors, $\mathbf{Y}$ is a diagonal matrix with labels of training samples on the diagonal. The dual is convex with a unique global optimum. By solving either primal or dual, one can obtain the both kernel and SVM parameters and thus kernel is learnt.

There are other versions of MKL [5, 83] in the literature. The main difference between all them is the difference in objective function formulation. One of the extension are *infinite kernel learning* [84] which combines the advantages of *kernel target alignment* and *multiple kernel learning* , allowing to learn liner kernel combination, while at the same time adjusting the kernel parameters. In this thesis we use and extend the above formulation.

2.7 Further challenges in kernel methods

There are many challenges in kernel methods. One of the major limitations in kernel methods is the complexity of training and testing process. So far we have seen kernel algorithms provide a boost in

performance by mapping the input samples to feature and then applying linear algorithm over there. And this mapping is done efficiently with the help of kernel function. Kernel methods access the feature space via the input samples and hence kernel algorithms need to store all the relevant input samples. For instance, testing in case of SVM for a new sample all the support vectors are needed to be stored so that they can be used to project on the separating hyperplane. The complexity of this testing process is high, as the size of kernel matrices increase quadratically as the number of SVs increases. So reduction of such complexities is highly necessary to run a particular set of applications faster. There are number of attempts to do this [85, 86], but still this is a challenging problem with scope for further research.

Another limitation is the appropriateness of choice of kernels. Unless the data is represented in appropriate feature space, improvement in performance of method cannot be seen. For this, researchers have started with designing the kernels and now it is moving towards "learning the kernel". In this thesis we work on later part and learn the kernel in nonlinear fashion rather than traditional linear fashion, and show the improvement in the performances at tasks like feature selection.

Kernel algorithms have brought a significant boost in the performance on the tasks like object recognition, object detection, object localization, etc. Some of kernel designs are seen in the chapter. In the case of images, the representations is much high dimensional, the limitation on the complexity holds here. The reduction of such complexities is still an active research area.

Chapter 3

Literature Survey on Kernel Learning

3.1 Overview

The performance of the learning algorithms for tasks like classification and regression strongly depends on the data representation. In kernel methods, the data representation is implicitly choosen through the kernel $k(\mathbf{x}, \mathbf{y})$. This kernel actually defines the similarity between two samples $\mathbf{x}, \mathbf{y}$, while defining an appropriate regularization term for the learning problem. In some situations, more flexible models are required. Recent works, show that using multiple kernels instead of a single one can enhance the interpretability of the decision function and improved performances.

Some of the earliest work on MKL was developed in [87, 88]. Their focus was on optimizing loss functions such as kernel target alignment rather than the specific classification or regression problem at hand. This was addressed in the influential work of [2] which showed how MKL could be formulated appropriately for a given task and optimized as an Semi-Definite Programming (SDP) or Quadratically Constrained Quadratic Programming (QCQP) for non-negative kernel weights. Nevertheless, QCQPs do not scale well to large problems and one of the first practical MKL algorithms was presented in [3]. In [3], the block l_1 formulation , in conjunction with M-Y regularization, was developed so that efficient gradient descent could be performed using the Sequential Minimizing Optimization (SMO) algorithm while still generating a sparse solution.

The work presented in [4] retained the block l_1 regularization and reformulated the problem as a Semi-Infinite Linear Programming problem (SILP). This made it applicable to large scale problems and the authors were impressively able to train their algorithm on a million splice data set. Further efficiency was obtained in [5,39] via gradient descent optimization and [83] opened up the possibility of training on an exponentially large number of kernels. Other interesting approaches have been proposed in [89–92] and include Hyper-kernels and multi-class MKL. In the next section the methods proposed in [2–5, 88] are discussed in detail and discuss how the problem has been tackled at the beginning to the latest methodology used for solving the problem.

3.2 MKL Approaches

3.2.1 Kernel Target Alignment [1]

This is the one of the initial paper which sought the idea of learning the kernel for improvement in the performance. It starts with the idea of defining the notion of the alignment between two kernels. This notion of alignment between two kernels is then extended to the alignment between kernel and labels, by constructing a "target kernel" $t(\mathbf{x_i}, \mathbf{x_j}) = y_i y_j$ with $y_i \in \{-1, +1\}$. The alignment is defined as

$$\begin{aligned} A(k,t) &= \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^t \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \mathbf{y}\mathbf{y}^t, \mathbf{y}\mathbf{y}^t \rangle_F}} \\ &= \frac{\langle \mathbf{K}, \mathbf{y}\mathbf{y}^t \rangle_F}{m\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F}} \end{aligned} \tag{3.1}$$

where $\mathbf{y}$ is vector of labels of the training samples and m is the number of training samples. Optimal kernel $\mathbf{K_{opt}}$ is characterized as below equation,

$$\mathbf{K_{opt}} = \sum_{l=1} d_l \mathbf{K}_l \tag{3.2}$$

where $\mathbf{K}_l = v_l v_l^t$ and v_l is eigen vectors of original kernel matrix $\mathbf{K}$. Now the optimal alignment becomes

$$A(\mathbf{K_{opt}}) = \frac{\langle \mathbf{K_{opt}}, \mathbf{y}\mathbf{y}^t \rangle_F}{m\sqrt{\sum_{ij} d_i d_j \langle v_i v_i^t, v_j v_j^t \rangle_F}} = \frac{\sum_l d_l \langle v_l, \mathbf{y} \rangle_F^2}{m\sqrt{\sum_l d_l^2}} \tag{3.3}$$

Maximizing the alignment by adding the constraint $\sum_l d_l^2 = 1$ yields the following Lagrangian formulation.

$$\max \quad \sum_l d_l \langle v_l, \mathbf{y} \rangle_F^2 - \lambda(\sum_l d_l^2 - 1) \tag{3.4}$$

Solving above gives $d_l \propto \langle v_l, \mathbf{y} \rangle_F^2$. It is shown that learning the kernel has improved the performance when compared to direct kernel. This has established the use of linear combinations of base kernels. Although, some generalization bounds have been given, the task is not directly related to classification and does not easily generalize to other loss functions. This is one of the major drawback of this method.

3.2.2 Learning the Kernel Matrix with SDP [2]

In [2], it is shown that Semi-Definite Programming (SDP) techniques can be applied to learn the kernel matrix. In specific, this work focuses on loss function for classification problem (in SVM framework). It involves joint optimization of kernel matrices and the coefficients in a conic combination of kernel matrices and the coefficients of a discriminative classifier. Finally, the problem is posed as a QCQP problem, which is special form of SDP.

This work apply the idea to the problem of combining data from multiple sources. Specifically, assuming that each source is associated with a kernel function, such that a training set yields a set of kernel matrices. The tools that they developed in their work made it possible to optimize over the coefficients in a linear combination of such kernel matrices. These coefficients can then be used to form linear combinations of kernel functions in the overall classifier. Thus, this approach allows us to combine heterogeneous data sources, making use of the reduction of heterogeneous data types to the common framework of kernel matrices, and choosing coefficients that emphasize those sources most useful in the classification decision. This later was named as Multiple Kernel Learning.

Semidefinite programming and Multiple kernel learning

Semidefinite programming deals with the optimization of convex functions over the convex cone of symmetric, positive semidefinite matrices

$$\mathcal{P} = \{X \in \mathbb{R}^{p \times p} \mid X = X^t,\ X \succeq 0\} \tag{3.5}$$

or affine subsets of this cone. With this, given $\mathcal{P}$ can be viewed as a search space for possible kernel matrices. This search space is constrained in order to prevent overfitting and achieve good generalization on test data. For MKL, a restricted set $\mathcal{K}$ of kernels is taken which is a set of positive semidefinite matrices. And these are bounded with trace that can be expressed as a linear combination of kernel matrices from the set $\{\mathbf{K}_1, \cdots, \mathbf{K}_l\}$. That is, $\mathcal{K}$ is the set of matrices $\mathbf{K}$ satisfying

$$\mathbf{K} = \sum_l d_l \mathbf{K}_l, \quad \mathbf{K} \geq 0, \quad trace(\mathbf{K}) \leq c \tag{3.6}$$

Additionally the parameters d_l can be constrained to be non-negative (i.e. $d_l \geq 0$). By doing so, a significant computational complexity is reduced. The problem is formulated as following quadratically constrained quadratic problem.

$$\begin{aligned} \min \quad & \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum_i^M \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^t\Phi_d(\mathbf{x_i}) + b) \geq 1 - \xi_i, \xi_i > 0 \quad \forall i \\ & \mathbf{K} = \sum_l d_l \mathbf{K}_l, \quad d_l \geq 0 \quad \forall l \\ & trace(\mathbf{K}) = c \end{aligned} \tag{3.7}$$

where c is a constant. This convex optimization problem, a QCQP more precisely, is a special instance of an SOCP (Second-Order Cone Programming problem, which can be solved efficiently), which is in turn a special form of SDP. Sparse kernel weights are obtained by solving the formulation. This can also be extended to optimise an appropriate cost function depending on the task at hand. Other possible loss functions are square hinge, KTA, regression, etc.

Solving the formulation in QCQP is more challenging than a Quadratic Programming (QP) problem, but in principle it can be solved by general-purpose optimization toolboxes. But QCQP does not scale well and becomes rapidly intractable as the number of learning examples or kernels become large.

3.2.3 MKL with Sequential Minimization Optimization Algorithm [3]

The formulation in SDP framework is convex but is a non-smooth minimization problem. This makes the direct application of simple local descent algorithms such as sequential minimization optimization infeasible. Therefore, [3] has considered the smoothed version of problem for which they proposed a SMO-like algorithm that enables to tackle medium-scale problems.

A classification algorithm called support kernel machine (SKM) was introduced in [3]. It is motivated as a block-based variant of the SVM and related to margin-based classification algorithms. But their underlying motivation was the fact that the dual of the SKM is exactly the problem (3.7) which they establish in their work. Here, input sample $\mathbf{x_i}$ in divided into n blocks and is represented as $\mathbf{x_i} = (\mathbf{x_{1i}}, \mathbf{x_{2i}}, \cdots, \mathbf{x_{ni}})$. For kernelization, mapped feature space of input sample is assumed to have n components $\Phi(\mathbf{x_i}) = (\Phi_1(\mathbf{x_i}), \cdots, \Phi_n(\mathbf{x_i}))$. Thus w also has the same block decomposition $\mathbf{w} = (\mathbf{w_1}, \cdots, \mathbf{w_n})$. SKM is then extended to SMO-like algorithm making use of Moreau-Yosida (MY) regularization. Finally the formulation is,

$$\begin{aligned} \min \quad & \frac{1}{2}(\sum_l d_l ||\mathbf{w_l}||_2)^2 + \frac{1}{2}\sum_l a_l^2 ||\mathbf{w_l}||_2^2 + C\sum_i \xi_i \\ \text{subject to} \quad & y_i(\sum_l \mathbf{w}_l^t \Phi_l(\mathbf{x_i}) + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned} \tag{3.8}$$

where (a_l) are the MY-regularization parameters. Block l_1-regularization ensures sparsity at block level and makes $\mathbf{w}$ to be sparse. This method has successfully enabled to tackle medium scale problems but not well to large scale data.

3.2.4 Large Scale MKL using SILP [4]

The approach in [4] reformulates the problem as semi-infinite linear program (SILP). This algorithm solves the problem iteratively solving a classical SVM problem with a single kernel and a linear program who's number of constraints increases along with iterations. This is one of major advantage of the method as there exists a lot of toolboxes to solve SVM with single kernel and thus tackles the problem with large-scale data. The formulation given in [3] is posed as following SILP program.

$$\begin{aligned} \max \quad & \theta \\ \text{subject to} \quad & d_l \geq 0, \ \sum_l d_l = 1 \\ & \sum_l d_l (\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k_l(\mathbf{x_i}, \mathbf{x_j}) - \sum_i \alpha_i) \geq \theta \quad \forall \boldsymbol{\alpha} \in \mathcal{C} \\ & \mathcal{C} = \{\boldsymbol{\alpha} \in \mathbb{R}^m | \ 0 \leq \alpha_i \leq C, \ \sum_i y_i \alpha_i = 0\} \end{aligned} \tag{3.9}$$

The above formulation is solved efficiently by using cutting plane method [93]. It is an iterative approach, in which the first step computes the optimal $(\mathbf{d}, \theta)$ for a restricted subset of constraints. Then

in the next step another algorithm generates a new, unsatisfied constraint determined by $\boldsymbol{\alpha}$. These constraints are added to the set of constraints in the first step and the iterations continue until the creteria $\sum_l d_l(\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k_l(\mathbf{x_i},\mathbf{x_j}) - \sum_i \alpha_i) \geq \theta$ is satisfied. First part is Linear Programming (LP) problem and second step can be solved through SVM. Essentially it is iterative LP-QP solution. In each iteration LP problems grows more complex as the constraint set increases. This algorithm can be extended to a large class of convex loss functions.

This solution can now tackle large scale problems and it is shown that it is capable of solving the problem consisting of 30,000 examples and 20 kernels in reasonable time. But, does not scale well to the problems which deals with large number of kernels.

3.2.5 Simple MKL [5]

Another algorithm which scales the solution of MKL problem to larger problems is presented in [5]. This method is based upon gradient descent optimization and obtains further efficiency when compared to SILP in scaling to large problems. The algorithm is fairly simple and uses following alternate optimization algorithm.

$$\min_d J(d) \quad \text{such that } \sum_{l=1} d_l = 1,\ d_l \geq 0 \tag{3.10}$$

where

$$J(d) = \begin{cases} \min & \frac{1}{2}\sum_l \frac{1}{d_l}||\mathbf{w_l}||^2 + C\sum_i \xi_i \quad \forall i \\ \text{s.t.} & y_i(\sum_l {\mathbf{w_l}}^t \Phi_l(\mathbf{x_i}) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \quad \forall i \end{cases} \tag{3.11}$$

Here, problem (3.10) is solved by using a simple gradient descent scheme. The objective function $J(d)$ is actually an optimal SVM objective value. This formulation results in a smooth and convex optimization problem which is equivalent to other MKL approaches discussed earlier. But the new objective function is more smoother, which makes descent methods practical.

Similar to the SILP algorithm, final algorithm iterates over two steps until some convergence is met. One step performs gradient descent to estimate kernel parameters, another step uses simple SVM to estimate classifier parameters. This is much efficient as the number of steps need for convergence is less when compared to SILP approach. This is because SILP approach does not use smoothness of the objective function. Other methods which are similar to this approach is [39, 83, 94, 95], the difference comes with change in objective function formulation.

3.2.6 Other Approaches

Besides these approaches, some other interesting approaches are infinite-dimensional kernel families such as hyper-kernels [90, 91] or general convex classes of kernels [89]. Other approaches aims at studing the regularization for sparsity in kernel selection [95–97]. There are also some approaches on extending to multi-class [92] and multi-label multiple kernel learning [98].

3.3 Remarks

All the methods described in the previous section essentially learn linear combinations of base kernels subject to l_1, or sometimes l_2 [88, 99], regularization of the kernel parameters. Most formulations are convex or can be made so by a change of variables. On the other hand, hierarchical multiple kernel learning [83] considers learning a linear combination of an exponential number of linear kernels, which is efficiently represented as a product of sums. This method can also be classified as learning a non-linear combination of kernels but the base kernels are restricted to concatenation kernels.

Chapter 4

Generalized Multiple Kernel Learning

4.1 Introduction

The success of SVMs at different tasks is often dependent on the choice of a good kernel and features – ones that are typically hand-crafted and fixed in advance. However, hand-tuning kernel parameters can be difficult as can selecting and combining appropriate sets of features. Multiple Kernel Learning (MKL) seeks to address this issue by learning the kernel from training data. In particular, it focuses on how the kernel can be learnt as a linear combination of given base kernels. Many MKL formulations have been proposed in the literature.

Nevertheless, MKL approaches are limited in that they focus on learning linear combinations of base kernels – corresponding to the concatenation of individual kernel feature spaces. Far richer representations can be achieved by combining kernels in other fashions. For example, taking products of kernels corresponds to taking a tensor product of their feature spaces. This leads to a much higher dimensional feature representation as compared to feature concatenation. Furthermore, by focusing mainly on feature concatenation, MKL approaches do not consider the fundamental question of what are appropriate feature representations for a given task. This can also be illustrated with an help of example. In the Figure 4.1, sample 1, 2 (in red color) belongs to one class and sample 3, 4 (in blue color) belongs to other class. A classifier which separate these classes cannot be built neither in individual feature spaces nor in combined kernel space using sum. But, by using kernel space of product of kernels, a classifier which can seperate both the classes can be built easily.

Here, we observe that it is fairly straight forward to extend traditional MKL formulations to handle generic kernel combinations. Furthermore, the gradient descent optimization developed and used in [5, 39, 83, 100] can still be applied out of the box. It is therefore possible to learn rich feature representations without having to sacrifice any of the advantages of a well developed, large scale optimization toolkit. In addition to the kernel function, it is also possible to generalize the regularization on the kernel parameters. This can be used to incorporate prior knowledge about the kernel parameters if available. However, the price that one has to pay for such generality, is that the new MKL formulation is no longer convex. Nevertheless, we feel that the ability to explore appropriate feature representation is probably more important than being able to converge to the global optimum (of an inappropriate representation).

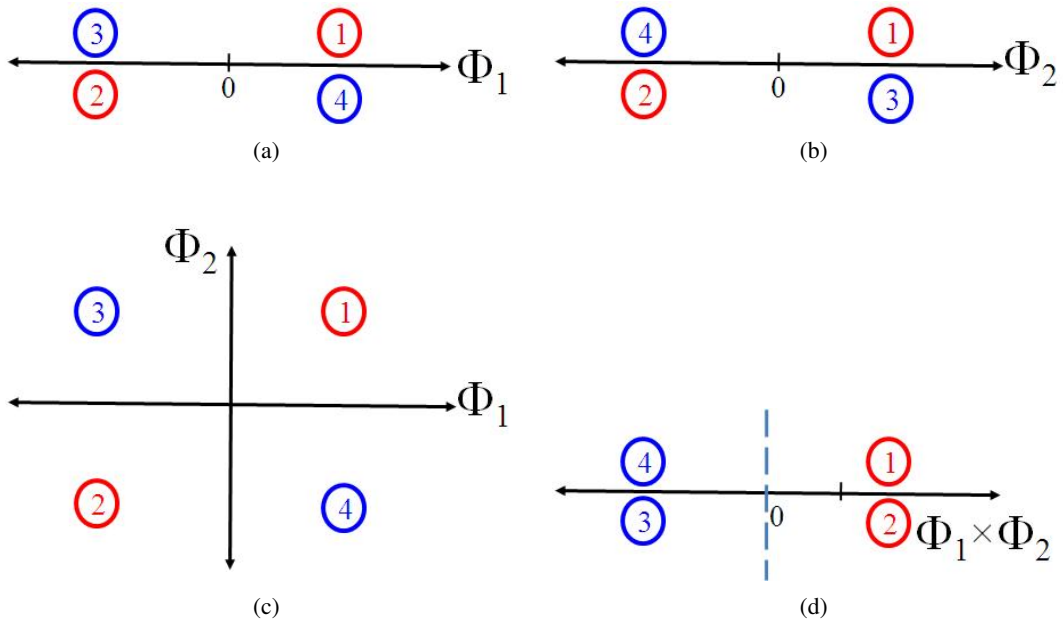


Figure 4.1: In (a), (b) data points are in individual 1-D feature spaces Φ_1, Φ_2. In (c), (d) data points are in combined kernel feature spaces, sum and product of kernels respectively. It can been seen that data points are not seperable in individual feature spaces and sum of kernel feature space. But, they are seperable in product of kernels space.

This is borne out by our experimental results.

In this chapter we present the details of GMKL in Section 4.2, 4.3 and we extend it for multi-class problem in Section 4.4 similar to the one proposed in [5] which will be useful for feature selection while extending to multi-class problems.

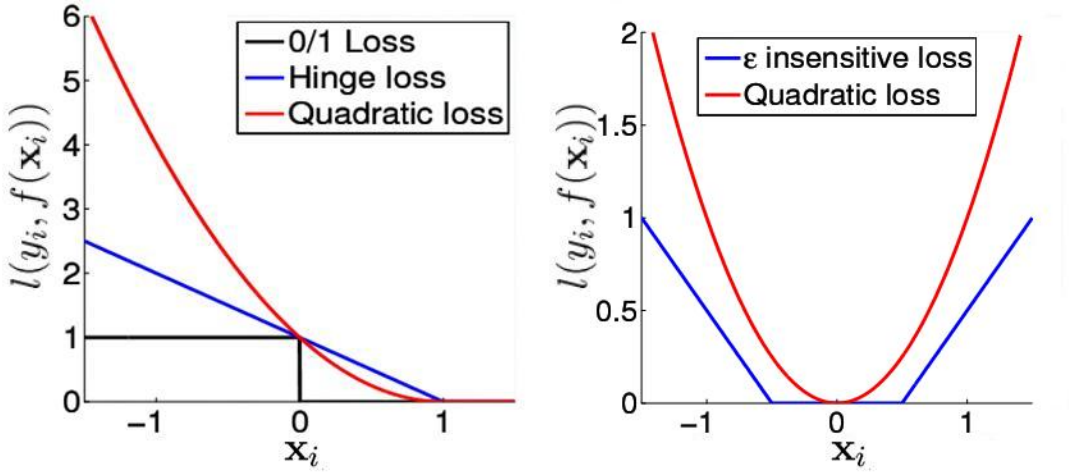
4.2 Generalized MKL : Formulation

Our objective is to learn a function of the form $f(\mathbf{x}) = \mathbf{w}^t \boldsymbol{\phi}_{\mathbf{d}}(\mathbf{x}) + b$ with the kernel $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}_{\mathbf{d}}^t(\mathbf{x}_i)\boldsymbol{\phi}_{\mathbf{d}}(\mathbf{x}_j)$ representing the dot product in feature space $\boldsymbol{\phi}$ parameterized by $\mathbf{d}$. The function can be used directly for regression or the sign of the function can be used for classification. The goal in SVM learning is to learn the globally optimal values of $\mathbf{w}$ and b from training data $\{(\mathbf{x}_i, y_i)\}$. In addition, MKL also estimates the kernel parameters $\mathbf{d}$. We extend the MKL formulation of [39] to

$$\min_{\mathbf{w},b,\mathbf{d}} \quad \tfrac{1}{2}\mathbf{w}^t\mathbf{w} + \sum_i l(y_i, f(\mathbf{x}_i)) + r(\mathbf{d}) \tag{4.1}$$

$$\text{subject to} \quad \mathbf{d} \geq 0 \tag{4.2}$$

where both the regularizer r and the kernel can be any general differentiable functions of $\mathbf{d}$ with continuous derivative. And l could be one of various loss functions such as $l = C \max(0, 1 - y_i f(\mathbf{x}_i))^p$ for classification or ϵ insensitive loss $l = C \max(0, |y_i - f(\mathbf{x}_i)| - \epsilon)$ for regression (see Figure 4.2). In case



(a) Classification (b) Regression

Figure 4.2: Commonly used (a) classification and (b) regression loss functions. For classification 0/1 loss function penalizes 1 for every misclassification. It is discontinuous and not convex where as hinge and quadratic are convex. For regression analysis ϵ insensitive and quadratic loss are used widely. Image courtsey [9]

of classification when p equals to one it becomes hinge loss and when p equals to two it is quadratic loss function. For regression other loss functions are quadratic loss $l = C(y_i - f(\mathbf{x}_i))^2$.

Three things are worth noting about the primal. First, we choose to use a non-convex formulation, as opposed to the convex $\sum_l \mathbf{w}_l^t \mathbf{w}_l / d_l$ [5], since for general kernel combinations, $\mathbf{w}_l^t \mathbf{w}_l$ need not tend to zero when d_l tends to zero. Second, we place $r(\mathbf{d})$ in the objective and incorporate a scale parameter within it rather than having it as an equality constraint (typically $\sum_l d_l = 1$ or $\sum_l d_l^2 = 1$). Third, the constraint $\mathbf{d} \geq 0$ can often be relaxed to a more general one which simply requires the learnt kernel to be positive definite. Conversely, the constraints can also be strengthened if prior knowledge is available. In either case, if $\nabla_{\mathbf{d}} r$ exists then the gradient descent based optimization is still applicable. However, the projection back into the feasible set can get more expensive.

4.3 Generalized MKL : Algorithm

In order to leverage existing large scale optimizers, we follow the standard procedure [100] of reformulating the primal as a nested two step optimization. In the outer loop, the kernel is learnt by optimizing over $\mathbf{d}$ while, in the inner loop, the kernel is held fixed and the SVM parameters are learnt. This can be achieved by rewriting the primal as follows

$$\begin{aligned} \min_{\mathbf{d}} \quad & T(\mathbf{d}) \quad \text{subject to} \quad \mathbf{d} \geq 0 \\ where \quad & T(\mathbf{d}) = \min_{\mathbf{w},b} \tfrac{1}{2}\mathbf{w}^t\mathbf{w} + \sum_i l(y_i, f(\mathbf{x}_i)) + r(\mathbf{d}) \end{aligned} \tag{4.3}$$

We now need to prove that $\nabla_{\mathbf{d}} T$ exists, and calculate it efficiently, if we are to utilize gradient

descent in the outer loop. This can be achieved by moving to the dual formulation of T given by (for classification and regression respectively)

$$W_C(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \quad \mathbf{1}^t\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^t\mathbf{Y}\mathbf{K}_\mathbf{d}\mathbf{Y}\boldsymbol{\alpha} + r(\mathbf{d}) \tag{4.4}$$

$$\text{subject to} \quad \mathbf{1}^t\mathbf{Y}\boldsymbol{\alpha} = 0, \;\; 0 \leq \boldsymbol{\alpha} \leq C \tag{4.5}$$

and

$$W_R(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \quad \mathbf{1}^t\mathbf{Y}\boldsymbol{\alpha} - \tfrac{1}{2}\boldsymbol{\alpha}^t\mathbf{K}_\mathbf{d}\boldsymbol{\alpha} + r(\mathbf{d}) - \epsilon\mathbf{1}^t|\boldsymbol{\alpha}| \tag{4.6}$$

$$\text{subject to} \quad 1^t\boldsymbol{\alpha} = 0, \;\; 0 \leq |\boldsymbol{\alpha}| \leq C \tag{4.7}$$

where $\mathbf{K}_\mathbf{d}$ is the kernel matrix for a given $\mathbf{d}$ and $\mathbf{Y}$ is a diagonal matrix with the labels on the diagonal.

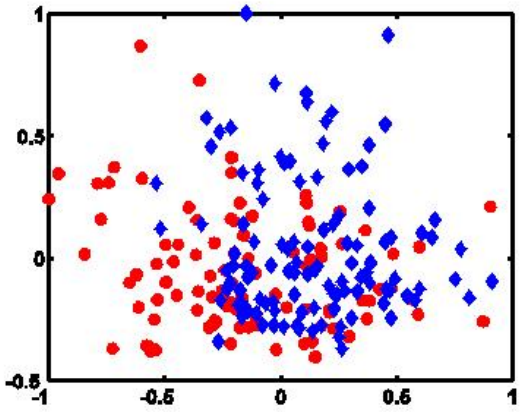
Note that we can write $T = r + P$ and $W = r + D$ with strong duality holding between P and D. Therefore, $T(\mathbf{d}) = W(\mathbf{d})$ for any given value of $\mathbf{d}$, and it is sufficient for us to show that W is differentiable and calculate $\nabla_\mathbf{d} W$. Proof of the differentiability of W_C and W_R comes from Danskin's Theorem [101]. Since the feasible set is compact, the gradient can be shown to exist if k, r, $\nabla_\mathbf{d} k$ and $\nabla_\mathbf{d} r$ are smoothly varying functions of $\mathbf{d}$ and if $\boldsymbol{\alpha}^*$, the value of $\boldsymbol{\alpha}$ that optimizes W, is unique. Furthermore, a straight forward extension of Lemma 2 in [100] can be used to show that W_C and W_R (as well as others obtained from loss functions for novelty detection, ranking, *etc.*) have derivatives given by

$$\frac{\partial T}{\partial d_l} = \frac{\partial W}{\partial d_l} = \frac{\partial r}{\partial d_l} - \tfrac{1}{2}\boldsymbol{\alpha}^{*t}\frac{\partial \mathbf{H}}{\partial d_l}\boldsymbol{\alpha}^* \tag{4.8}$$

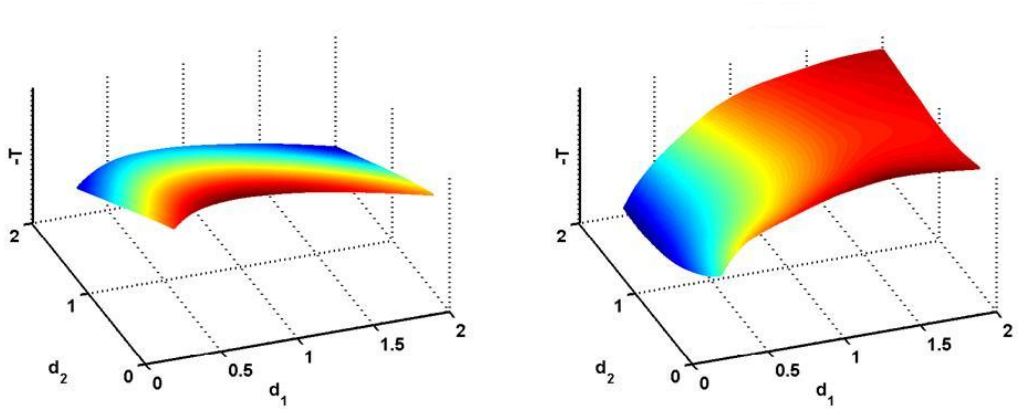
where $\mathbf{H} = \mathbf{Y}\mathbf{K}\mathbf{Y}$ for classification and $\mathbf{H} = \mathbf{K}$ for regression. Thus, in order to take a gradient step, all we need to do is obtain $\boldsymbol{\alpha}^*$. Note that since W_C or W_R are equivalent to their corresponding single kernel SVM duals with kernel matrix $\mathbf{K}_\mathbf{d}$, $\boldsymbol{\alpha}^*$ can be obtained by any SVM optimization package. The final algorithm is given in Algorithm 1 and we refer to it as Generalized MKL (GMKL). The step size s^n is chosen based on the Armijo rule to guarantee convergence and the projection step, for the constraints $\mathbf{d} \geq 0$, is as simple as $\mathbf{d} \leftarrow \max(\mathbf{0}, \mathbf{d})$. Note that the algorithm is virtually unchanged from [39]

Algorithm 1 Generalized MKL.

1: $n \leftarrow 0$
2: Initialize $\mathbf{d}^0$ randomly.
3: **repeat**
4: $\quad \mathbf{K} \leftarrow k(\mathbf{d}^n)$
5: $\quad$ Use an SVM solver of choice to solve the single kernel problem with kernel $\mathbf{K}$ and obtain $\boldsymbol{\alpha}^*$.
6: $\quad d_l^{n+1} \leftarrow d_l^n - s^n \left(\frac{\partial r}{\partial d_l} - \frac{1}{2}\boldsymbol{\alpha}^{*t}\frac{\partial \mathbf{H}}{\partial d_l}\boldsymbol{\alpha}^* \right)$
7: $\quad$ Project $\mathbf{d}^{n+1}$ onto the feasible set if any constraints are violated.
8: $\quad n \leftarrow n + 1$
9: **until** converged



(a)



(b) (c)

Figure 4.3: (a) Plot of UCI dataset (Sonar) using first two dimensions. (b) Value of objective function using sum of kernels. (c) Value of objective function using product of kernels.

apart from the more general form of the kernel k and regularizer r. If a faster rate of convergence was required, our assumptions could be suitably modified so as to take second order steps rather than perform gradient descent.

Only very mild restrictions have been placed on the form of the learnt kernel k and regularizer r. As regards k, the only constraints that have been imposed are that $\mathbf{K}$ be strictly positive definite for all valid $\mathbf{d}$ and that $\nabla_{\mathbf{d}} k$ exists and be continuous. Many kernels can be constructed that satisfy these properties. One can, of course, learn the standard sum of base kernels. More generally, products of base kernels, and other combinations which yield positive definite kernels, can also be learnt now. In addition, one can also tune kernel parameters in general kernels such as $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = (d_0 + \sum_l d_l \mathbf{x}_i^t \mathbf{A}_l \mathbf{x}_j)^n$ or $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\sum_l d_l \mathbf{x}_i^t \mathbf{A}_l \mathbf{x}_j}$. Combined with a sparsity promoting regularizer on $\mathbf{d}$, this can be used for non-linear dimensionality reduction and feature selection for appropriate choices of $\mathbf{A}$. Note, however, that such kernels do not lead to convex formulations.

As regards r, we only require that its derivative should exist and be continuous. Since $\mathbf{d}$ can be

restricted to the non-negative orthant, various forms of p-norm regularisers with $p \geq 1$ fall in this category. In particular, l_1 regularization with $r(\mathbf{d}) = \boldsymbol{\sigma}^t\mathbf{d}$ or variations of [12] could be used for learning sparse solutions. Alternatively, l_2 regularization of the form $r(\mathbf{d}) = (\mathbf{d} - \boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{d} - \boldsymbol{\mu})$ can be used when only a small number of relevant kernels are present or if prior knowledge in the form of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is available (for instance, from transfer learning). Finally, for regression, one could incorporate the term $\log|K_{\mathbf{d}}|$ into r so as to obtain a MAP estimate of a probabilistic formulation. The naïve substitution $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}$ would then render our formulation near identical to a marginal likelihood approach in Gaussian Processes.

Toy Example : For toy dataset we took Sonar dataset from UCI repository. From this we picked two features (see Figure 4.3a). Plots of the objective functions corresponding to sum and product of kernels is given in 4.3b and 4.3c respectively. Although the product of kernels is a non-convex function the objective function is smooth which enables to do gradient descent search.

4.4 Multi-class extensions

The proposed GMKL for binary classification problem can be extended to tackle multi-class classification problems. Multi-class classifiers aims to assign labels to instances using learnt model, where the labels are drawn from a finite set of several elements. The most common approach is to reduce the single multi-class problem into multiple binary classification problems. Each of the problems yield a binary classifier, which produce an output function that gives relatively large values for examples from the positive class and relatively small values for examples from the negative class. There are two common methods to build such binary classifiers, where each classifier distinguishes between (i) one of the class labels to the rest (one-versus-all) or (ii) between every possible pair of classes (one-versus-one).

Consider there are N classes in a multi-class classification problem. Classifiers can be built in one of the following two approaches :

1. One-versus-All : Here N binary classifiers are built, where i^{th} classifier is trained with the examples in the i^{th} class as positive labels and others as negative labels. For classifying unseen samples out of the N classifiers, the classifier with the highest output assigns the class.

2. One-versus-One : Here classifiers are built for every possible pair of classes. So $N(N-1)/2$ classifiers are built. For new instances, max-wins voting strategy is used. In which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with maximum votes is assigned to it.

In the two approaches a different set of kernel weights was learnt for each class. Above two approaches are widely used and performs well in many practical applications.

Other approach than splitting into multi-classifiers is to optimize all the classifiers altogether. We also adopt the multi-class parameter sharing strategy of [5] and extend our generalized kernel learning

framework to the following N class primal

$$\underset{\mathbf{w}_n, b_n, \mathbf{d}, \boldsymbol{\xi}_n}{\text{Min}} \quad l(\mathbf{d}) + \sum_{n=1}^{N} \tfrac{1}{2}\mathbf{w}_n^t \mathbf{w}_n + C\mathbf{1}^t \boldsymbol{\xi}_n \tag{4.9}$$

$$\text{subject to} \quad \mathbf{Y}_n(\boldsymbol{\phi}_n^t \mathbf{w}_n + b_n) \geq \mathbf{1} - \boldsymbol{\xi}_n \tag{4.10}$$

$$\boldsymbol{\xi}_n \geq 0, \;\; \mathbf{d} \geq 0 \tag{4.11}$$

where the subscript n refers to the n^{th} 1-vs-All problem. Applying the standard nested optimization strategy yields the gradient direction

$$\frac{\partial T}{\partial d_l} = \frac{\partial W}{\partial d_l} = \frac{\partial l}{\partial d_l} - \tfrac{1}{2}\sum_{n=1}^{N} \boldsymbol{\alpha}_n^{*t} \mathbf{Y}_n \frac{\partial \mathbf{K}_n}{\partial d_l} \mathbf{Y}_n \boldsymbol{\alpha}_n^{*} \tag{4.12}$$

where α_n^* are the support vector coefficients for the n^{th} 1-vs-All SVM and $\mathbf{Y}_n$ and $\mathbf{K}_n$ are the corresponding training label and kernel matrices respectively. In summary, the problem is made tractable by optimizing all the $\boldsymbol{\alpha}$s independently, though they are all optimized jointly with $\mathbf{d}$. This particular formulation will be useful when we do tasks like feature selection. Other multi-class MKL formulations can be found in [92, 102].

4.5 Summary

We have proposed MKL can be extended to generic kernel combinations subject to general regularizations on kernel parameters. And we explained how it is done with out compromising with efficiency. We have shown how it can be applied to multi-class problems. In the next chapter, we show some of the applications of proposed method.

Chapter 5

Applications

5.1 Introduction

The GMKL formulation which we presented in the previous chapter is quite general and can be used for kernel combination, kernel parameter tuning, non-linear feature selection, dimensionality reduction and other applications which are listed in Chapter 1. In this thesis, we focus on following two applications (i) Feature Selection and (ii) Learning discriminative parts for object categorization. In both the applications we exploit the weights learnt during MKL.

Feature selection is an area of pattern recognition which is used to select a subset of relevant features for building robust learning models. The basic idea of feature selection is to remove most irrelevant and redundant features from the data, it helps to improve the performance of learning models by (i) Enhancing generalization capability. (ii) Improves the speed of learning process. (iii) Improving model interpretability. For our experiments, we employ UCI datasets [103] which are popular for benchmarking machine learning algorithms. In our experiments, we show that for a fixed number of selected features, standard MKL can lag behind our formulation. Stated in another way, our formulation is capable of reaching the same classification accuracy as MKL with only a sixth of the features. We also present comparative results with AdaBoost, OWL-QN [11], LP-SVM [13], Sparse SVM [12] and BAHSIC [104].

In the second application, our objective is to determine minimal sets of pixels and image regions required for a given object categorization task. Information present in images can be redundant and, therefore, looking at the entire image might not be necessary for performing certain classification tasks. In other cases, some image parts might influence decision making but might not be crucial. Such parts could potentially be ignored while still keeping classification accuracy above an acceptable threshold. Selecting discriminative pixels and regions can directly improve efficiency and compression and reduce data transmission costs. It can also be used to enhance our understanding of the object categorization problem at hand, determine the importance of context and highlight artifacts in the training data. We explore the use of multiple kernel learning to select the most relevant pixels and regions for classification. Results are presented on benchmark problems such as gender identification and object recognition on the Caltech-101 [14] and Caltech-256 databases [7].

In the following section we first discuss some of the popular feature selection methods which we compare with our method. Then we give details of the other experiments which compare with other other multiple kernel learning methods. In the section 5.3 we explain the problem of learning discriminative parts in detail and present related experiments.

5.2 Feature Selection

5.2.1 Popular Methods

As an application we apply Generalized MKL on feature selection problem. We compare our formulation to traditional MKL as well as the following feature selection methods

Boosting : The LPBoost formulation of [105] is similar to that of standard MKL and boosting generalizes standard MKL's decision function. Boosting can therefore be used to learn linear combinations of base kernels. Individual "weak classifier" SVMs are pre-learnt from each of the given base kernels and combined using AdaBoost. This can be attractive when there are a large number of kernels or when kernels are made available incrementally. While the computational costs are low, the empirical results were found to be poor as the learnt kernel weights could not influence the pre-learnt weak classifiers. Of course, in traditional boosting, the weak classifiers and the weights are learnt together and we present comparative results to [106] which represents a state-of-the-art boosting method for gender identification.

OWL-QN [11] : This is a large scale implementation of l_1 logistic regression. The method learns a function of the form $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x}$ by minimizing $(1/C)||w||_1 + \sum_i l(y_i, f(\mathbf{x}_i))$ where l is the log loss. Despite being linear, the method can sometimes outperform boosting. Nevertheless, the overall performance is poor as compared to the other linear methods since OWL-QN does not have an explicit bias term. One could simulate a bias by adding a constant feature but the corresponding weight could be set to zero due to the l_1 regularization. When this doesn't happen, OWL-QN's performance is comparable to LP-SVM and Sparse-SVM.

LP-SVM [13] : This is the standard SVM formulation but with the l_2 regularization on $\mathbf{w}$ replaced by l_1 regularization. We consider the linear formulation which learns a function of the form $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x}+b$ by minimizing $||w||_1 + C\sum_i l(y_i, f(\mathbf{x}_i))$ where l is the hinge loss. Seeing that the hinge loss is very similar to the log loss, the formulation appears to be very similar to OWL-QN. However, due to the explicit bias term b which is not included in the l_1 regularization, LP-SVM can sometimes perform much better than OWL-QN. Somewhat surprisingly, the performance of the linear LP-SVM could even be better than that of non-linear MKL (though not GMKL).

Sparse-SVM [12] : This method does not use explicit l_1 regularization to enforce sparsity. Instead, it places a direct cardinality constraint on the hyperplane normal. It learns a function of the form $f(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + b$ by minimizing $||w||_2 + C\sum_i l(y_i, f(\mathbf{x}_i))$ subject to $||\mathbf{w}||_0 \leq r$, where l is the hinge loss. We take the convex QCQP relaxation (QCQP-SSVM) proposed by the authors. Empirically, we found the performance of Sparse-SVM to be very similar to that of LP-SVM though, being a QCQP, it took much longer to optimize.

BAHSIC [104] : This is a leading filter method which runs a backward selection algorithm discarding features based on their label dependence as measured by the Hilbert-Schmidt independence criterion. We use an RBF kernel for the data (the same as used by boosting, MKL and GMKL) and a linear kernel for the labels. BAHSIC outputs a ranked list of features from which a subset of the desired size can be selected. An SVM with an RBF kernel is then trained using the selected features. In our experiments, we found backward selection to be computationally very expensive without offering any advantages in terms of classification accuracy. Given identical kernels, BAHSIC performed substantially worse than GMKL,

5.2.2 Experiments - UCI Datasets

In this section we evaluate generalized kernel learning on feature selection problems. We investigate this problem on the UCI datasets. We found out there can be as much as a 6% to 10% difference in performance between GMKL and MKL. We also demonstrate that GMKL performs better than the other methods considered.

To generate feature selection results, we can vary the hyper-parameter C in the wrapper methods to select a desired number of features. However, this strategy does not yield good classification results even though globally optimal solutions are obtained. Low values of C encouraged greater sparsity but also permitted more classification errors. We obtained much better results by the theoretically suboptimal strategy of fixing C to a large value (chosen via cross-validation so as to minimize classification error), learning the classifier, taking the top ranked components of $\mathbf{w}$ (or $\mathbf{d}$) and relearning the classifier using only the selected features.

This technique was used to generate the results in Tables 5.1 For each dataset, the very last row summarizes the number of features selected (N_s) by each wrapper method and the resultant classification accuracy. When the number of desired features (N_d) is less than N_s, the classification accuracy is determined using the N_d top ranked features. Otherwise, when $N_d > N_s$, the table entry is left blank as the classification accuracy either plateaus or decreases as suboptimal features are added. In such a situation, it is better to choose only N_s features and maintain accuracy.

For experiments on UCI datasets, we follow the standard experimental methodology [5] where 70% of the points are used for training and the remaining 30% for testing. We use 10% of the training data for validation. Results are reported over 20 random splits of the data. All datasets are preprocessed to have zero mean and unit variance. An RBF kernel is assigned to each of the M features in a given dataset. The M RBF kernels are then combined linearly for standard MKL and by taking their product for GMKL. The learnt kernels are of the form $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{M} d_l e^{-\gamma_l (x_{il} - x_{jl})^2}$ and $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_{l=1}^{M} e^{-d_l (x_{il} - x_{jl})^2}$ respectively.

Table 5.1 lists the feature selection results. AdaBoost tended to perform the worst and selected only few features on average. The poor performance was due to the fact that each of the SVMs was learnt independently. The weak classifier coefficients (i.e. kernel weights) did not influence the individual SVM parameters. By contrast, there is a very tight coupling between the two in MKL and GMKL and this ensures better performance.

LP-SVM, Sparse-SVM and OWL-QN perform even better than standard MKL (though not better than

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
5	75.2 ± 6.9	84.0 ± 6.0	86.7 ± 3.1	87.0 ± 3.1	87.1 ± 3.6	85.1 ± 3.2	$\mathbf{90.9 \pm 1.9}$
10	–	87.6 ± 2.2	90.6 ± 3.4	90.2 ± 3.5	90.2 ± 2.6	87.8 ± 2.4	$\mathbf{93.7 \pm 2.1}$
15	–	89.1 ± 1.9	93.0 ± 2.1	91.9 ± 2.0	92.6 ± 3.0	87.7 ± 2.2	$\mathbf{94.1 \pm 2.1}$
20	–	89.2 ± 1.8	92.8 ± 3.0	92.4 ± 2.5	93.4 ± 2.6	87.8 ± 2.8	–
25	–	89.1 ± 1.9	92.6 ± 2.7	92.4 ± 2.7	94.0 ± 2.2	87.9 ± 2.7	–
30	–	–	92.6 ± 2.6	92.9 ± 2.5	94.3 ± 1.9	–	–
34	–	–	92.6 ± 2.6	92.9 ± 2.5	$\mathbf{94.6 \pm 2.0}$	–	–
	75.1 (9.8)	89.2 (25.2)	92.6 (34.0)	92.9 (34.0)	–	88.1 (29.3)	94.4 (16.9)

(a) Ionosphere, $N = 246$, $M = 34$, Uniform MKL $= 89.9 \pm 2.5$, Uniform GMKL $= 94.6 \pm 2.0$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
3	79.4 ± 6.5	81.7 ± 2.7	76.4 ± 4.5	76.1 ± 5.8	85.2 ± 3.8	83.7 ± 4.4	$\mathbf{86.3 \pm 4.1}$
7	–	82.6 ± 3.3	86.2 ± 2.7	86.1 ± 4.0	88.5 ± 3.6	84.7 ± 5.2	$\mathbf{92.6 \pm 2.9}$
11	–	83.5 ± 2.8	86.0 ± 3.5	86.1 ± 3.1	89.4 ± 3.6	86.3 ± 4.3	–
15	–	–	87.0 ± 3.3	86.3 ± 3.1	89.9 ± 3.5	–	–
22	–	–	87.2 ± 3.2	87.2 ± 3.0	91.0 ± 3.5	–	–
	80.2 (5.2)	83.6 (11.1)	87.2 (22.0)	87.2 (22.0)	–	88.3 (14.6)	92.7 (9.0)

(b) Parkinsons, $N = 136$, $M = 22$, Uniform MKL $= 87.3 \pm 3.9$, Uniform GMKL $= 91.0 \pm 3.5$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
10	64.2 ± 4.0	72.8 ± 2.9	69.8 ± 5.1	72.6 ± 3.7	76.5 ± 3.5	80.0 ± 3.0	$\mathbf{81.1 \pm 3.8}$
20	65.5 ± 4.1	76.0 ± 4.4	73.8 ± 4.9	76.7 ± 4.1	83.6 ± 3.3	84.5 ± 3.4	$\mathbf{89.9 \pm 2.3}$
30	65.4 ± 4.1	80.8 ± 2.5	79.0 ± 2.8	79.4 ± 3.0	86.7 ± 2.8	86.2 ± 3.3	$\mathbf{92.6 \pm 1.7}$
40	–	81.6 ± 2.9	81.5 ± 3.2	81.8 ± 2.8	87.4 ± 2.8	87.0 ± 3.2	$\mathbf{93.3 \pm 2.0}$
60	–	83.0 ± 1.9	83.6 ± 2.8	83.5 ± 2.4	90.0 ± 2.6	87.8 ± 3.3	–
100	–	–	83.4 ± 2.9	83.3 ± 2.5	$\mathbf{93.6 \pm 1.8}$	–	–
166	–	–	83.4 ± 2.9	83.3 ± 2.5	$\mathbf{93.8 \pm 1.9}$	–	–
	65.5 (31.1)	83.5 (86.7)	83.4 (166.0)	83.3 (166.0)	–	88.2 (73.2)	93.6 (57.9)

(c) Musk, $N = 333$, $M = 166$, Uniform MKL $= 90.2 \pm 3.2$, Uniform GMKL $= 93.8 \pm 1.9$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
5	64.6 ± 6.6	68.9 ± 5.6	68.0 ± 7.9	68.4 ± 6.2	61.1 ± 6.6	70.4 ± 4.5	$\mathbf{74.4 \pm 5.1}$
10	67.9 ± 6.4	68.7 ± 4.6	71.5 ± 5.4	70.9 ± 5.9	73.1 ± 6.1	74.6 ± 5.6	$\mathbf{80.2 \pm 4.9}$
15	67.3 ± 6.4	71.4 ± 3.6	71.4 ± 3.3	72.2 ± 4.5	74.7 ± 7.7	76.5 ± 7.0	$\mathbf{80.7 \pm 5.5}$
20	–	73.1 ± 2.6	73.7 ± 2.8	74.0 ± 3.1	77.9 ± 5.7	79.5 ± 4.6	$\mathbf{82.0 \pm 5.3}$
25	–	73.5 ± 2.8	74.1 ± 3.5	73.6 ± 3.6	78.6 ± 5.2	81.1 ± 4.2	–
30	–	73.9 ± 3.2	73.4 ± 3.4	73.8 ± 3.9	80.8 ± 4.7	81.4 ± 4.2	–
40	–	–	73.6 ± 3.8	73.7 ± 3.8	81.4 ± 3.9	–	–
60	–	–	73.6 ± 3.6	73.5 ± 4.0	$\mathbf{84.6 \pm 4.1}$	–	–
	67.38 (18.7)	74.7 (39.3)	73.6 (60.0)	73.5 (60.0)	–	81.4 (38.6)	82.3 (20.4)

(d) Sonar, $N = 145$, $M = 60$, Uniform MKL $= 82.9 \pm 3.4$, Uniform GMKL $= 84.6 \pm 4.1$

N_d	AdaBoost	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
5	$\mathbf{76.7 \pm 2.2}$	74.2 ± 4.1	75.4 ± 2.7	75.5 ± 2.7	76.6 ± 2.1	67.8 ± 6.0	76.1 ± 3.8
10	–	77.2 ± 5.2	75.9 ± 4.6	76.5 ± 3.8	77.3 ± 2.3	68.7 ± 3.3	$\mathbf{77.8 \pm 3.3}$
15	–	77.8 ± 5.5	76.2 ± 5.1	77.2 ± 5.0	76.2 ± 0.0	69.4 ± 5.1	$\mathbf{78.3 \pm 3.6}$
20	–	78.1 ± 5.3	78.2 ± 5.2	77.7 ± 5.2	77.3 ± 6.3	70.1 ± 5.1	–

Database	SimpleMKL	GMKL
Sonar	80.6 ± 5.1 (793)	$\mathbf{82.3 \pm 4.8}$ (**60**)
Wpbc	76.7 ± 1.2 (442)	$\mathbf{79.0 \pm 3.5}$ (**34**)
Ionosphere	91.5 ± 2.5 (442)	$\mathbf{93.0 \pm 2.1}$ (**34**)
Liver	65.9 ± 2.3 (091)	$\mathbf{72.7 \pm 4.0}$ (**06**)
Pima	76.5 ± 2.6 (117)	$\mathbf{77.2 \pm 2.1}$ (**08**)

Table 5.2: Comparison with the results in [5]. GMKL achieves slightly better results but takes far fewer kernels as input.

Database	N	M	HKL	GMKL
Magic04	1024	10	84.4 ± 0.8	$\mathbf{86.2 \pm 1.2}$
Spambase	1024	57	91.9 ± 0.7	$\mathbf{93.2 \pm 0.8}$
Mushroom	1024	22	99.9 ± 0.2	$\mathbf{100 \pm 0.0}$

Table 5.3: Comparison between HKL and GMKL.

GMKL). BAHSIC, which is a non-linear filter method, follows the same trend and its performance is significantly worse than GMKL with identical kernels. This would suggest that wrapper methods based on the right feature representation should be preferable to filter methods which do not directly optimize for classification accuracy. For a fixed number of features, GMKL has the best classification results even as compared to MKL or BAHSIC (though the variance can be high for all the methods). Furthermore, a kernel with fixed uniform weights yields ballpark classification accuracies though GMKL can achieve the same results using far fewer features.

Comparison to SimpleMKL and HMKL

The classification performance of standard MKL can be improved by adding extra base kernels which are either more informative or help better approximate a desired kernel function. However, this can lead to a more complex and costlier learning task. We therefore leave aside feature selection for the moment and compare our results to those reported in [5]. Table 5.2 lists classification performance on the 5 datasets of [5]. The number of kernels input to each method are reported in brackets. As can be seen, GMKL can achieve slightly better performance than SimpleMKL while training on far fewer kernels. Of course, one can reduce the number of kernels input to SimpleMKL but this will result in reduced accuracy. In the limit that only a single kernel is used per feature, we will get back the results of Table 5.1 where GMKL does much better than standard MKL.

The results for Liver were obtained using l_2 regularization. This lead to a 7% improvement in performance over SimpleMKL as a sparse representation is not suitable for this database (there are only 6 features). Pima also has very few features but l_1 and l_2 regularization give comparable results.

Finally, we also compare results to Hierarchical MKL [83]. We use a quadratic kernel of the form $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = (1 + \sum_l d_l x_{il} x_{jl})^2$ as compared to the more powerful $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_l (1 + x_{il} x_{jl})^4$ of [83] which decomposes to a linear combination of 5^M kernels. Nevertheless, as shown in Table 5.3,

we achieve slightly better results with less computational cost.

5.3 Learning discriminative parts for object categorization

Our objective is to perform object categorization by focusing on only a subset of the pixels or regions present in an image. Information present in images can be redundant and looking at the entire image might not be necessary for performing certain classification tasks. Furthermore, even though some image parts might influence decision making, they might not be crucial. Such parts could potentially be ignored while still keeping the classification accuracy above an acceptable threshold.

Selecting discriminative image pixels can benefit many potential applications. It can improve the efficiency of object recognition algorithms and lead to better image compression. For instance, many mobile and network camera applications require transmitting images to a server for classification. By detecting and transmitting only the discriminative image regions one can enhance valuable battery life or reduce bandwidth consumption while paying only a small performance hit in terms of classification accuracy. In other areas such as astronomy and medical imaging, there is great interest in designing specialized cameras which take only a few image measurements rather than capturing the whole scene. For instance, in MRI [107], one might be willing to slightly sacrifice classification accuracy in order to gain imaging speed by making as few discriminative image measurements as possible. Selecting discriminative regions can also be used to enhance our understanding of the object categorization problem at hand, determine the importance of context and highlight artifacts in the training data.

Here, we use Multiple Kernel Learning (MKL) to select the most relevant pixels and regions for classification. Our goal is to select as few of these regions as possible while minimizing the impact on classification performance. Three scenarios are investigated depending on the form of the data. First, when the data is perfectly aligned, a kernel is associated with each pixel in the image and MKL is used to perform kernel (pixel) selection. Second, when the data has only rough alignment, the image is partitioned into a grid and a kernel associated with each grid element followed by MKL selection. Finally, when there can be no alignment, a kernel is associated with each codeword (in a bag of words framework) and MKL used to learn the most discriminative codebook entries. Only the image regions corresponding to the selected codewords are then kept. We employ FERET, Caltech 101 and Caltech 256 datasets to demonstrate the three scenarios respectively. Figures 5.2, 5.4 and 5.7 illustrate the three cases.

5.3.1 Related Work

One can use interest point detectors [108] to select "important" or "informative" regions in an image. Unfortunately, these regions are not learnt to maximize classification performance on the given task but are designed to maximize stability and repeatability. The method of [109] does learn object-specific salient parts for classification. However, it is geared towards object *identification* from very little positive training data rather than object categorization. A per image distance function is learnt for retrieval in [110, 111]. The algorithm learns weights for patch based image features and can be used to identify the salient image regions. However, our method selects pixels and regions at the category, rather than

N_d	AdaBoost	B&R 2007	OWL-QN	LP-SVM	S-SVM	BAHSIC	MKL	GMKL
10	76.3 ± 0.9	79.5 ± 1.9	71.6 ± 1.4	84.9 ± 1.9	79.5 ± 2.6	81.2 ± 3.2	80.8 ± 0.2	$\mathbf{88.7 \pm 0.8}$
20	–	82.6 ± 0.6	80.5 ± 3.3	87.6 ± 0.5	85.6 ± 0.7	86.5 ± 1.3	83.8 ± 0.7	$\mathbf{93.2 \pm 0.9}$
30	–	83.4 ± 0.3	84.8 ± 0.4	89.3 ± 1.1	88.6 ± 0.2	89.4 ± 2.4	86.3 ± 1.6	$\mathbf{95.1 \pm 0.5}$
50	–	86.9 ± 1.0	88.8 ± 0.4	90.6 ± 0.6	89.5 ± 0.2	91.0 ± 1.3	89.4 ± 0.9	$\mathbf{95.5 \pm 0.7}$
80	–	88.9 ± 0.6	90.4 ± 0.2	–	90.6 ± 1.1	92.4 ± 1.4	90.5 ± 0.2	–
100	–	89.5 ± 0.2	90.6 ± 0.3	–	90.5 ± 0.2	94.1 ± 1.3	91.3 ± 1.3	–
150	–	91.3 ± 0.5	90.3 ± 0.8	–	90.7 ± 0.2	94.5 ± 0.7	–	–
252	–	93.1 ± 0.5	–	–	90.8 ± 0.0	94.3 ± 0.1	–	–
	76.3 (12.6)	–	91 (221.3)	91 (58.3)	90.8 (252)	–	91.6 (146.3)	95.5 (69.6)

Table 5.4: Gender identification results. The final row summarizes the average number of features selected (in brackets) by each wrapper method and the resultant classification accuracy. See text for details.

the image, level. These are learnt directly for the classification task at hand. Finally, a lot of work has been done on image saliency which is related to our work (see [112, 113] and references within).

In the first scenario we use MKL as a feature selection tool for the task. So we compare with the methods described in section 5.2 and performance of MKL to these methods in Section 5.3.2.

5.3.2 Experiments

In this section we evaluate our generalized kernel learning formulation. To test the case when the images can be aligned we assess gender identification performance on the benchmark FERET subset of faces [10]. It is shown that the products of kernels formulation can achieve more than 95% classification accuracy by sampling as few as 30 pixels in an image. Thus, we can get an eight times compression factor while sacrificing less than half a percent in classification accuracy. These results are significantly better than the standard MKL formulation which achieves only 86.32% when restricted to 30 pixels. Similarly, for the unaligned case in some of the Caltech 256 classes, we can obtain nearly a three times compression factor while reducing classification accuracy by only 1%. Again, products of kernels are better than sums of kernels by nearly 10 to 15% for a fixed feature set size. However, for the roughly aligned case on Caltech 101 sums of kernels yield better results than products of kernels.

Gender Identification

We tackle the binary gender identification problem on the benchmark database of [10]. The database has images of 1404 male and 711 female faces giving a total of 1755 images in all (figure 5.1 shows sample images). We follow the experimental setup of [10] and use 1053 images for training and 702 for testing. Results are reported after averaging over 3 random splits of the training and test sets.

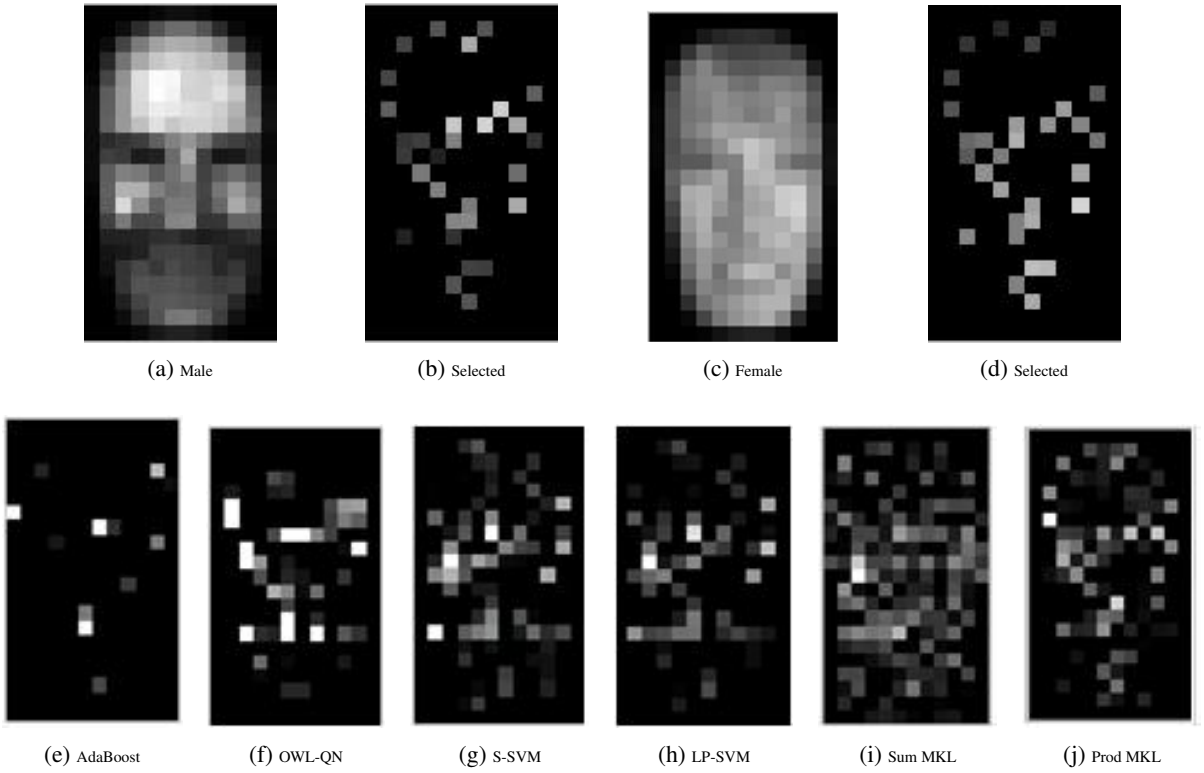
Each image in the database has been pre-processed by [10] to be aligned and has been scaled down to have dimensions 21×12. Thus, each image has 252 pixels and we associate an RBF kernel with each pixel based on its grey scale intensity directly. For Generalized MKL, the 252 base kernels are

combined by taking their product to get $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_{l=1}^{252} e^{-d_l(x_{il}-x_{jl})^2}$ where x_{il} and x_{jl} represent the intensity of the l^{th} pixel in image i and j respectively. For standard MKL, the same 252 base kernels are combined using the sum representation to get $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{252} d_l e^{-\gamma_m(x_{il}-x_{jl})^2}$. Both methods are subject to l_1 regularization on $\mathbf{d}$ so that only a few kernels are selected. AdaBoost can also be used to combine the 252 weak classifiers derived from the individual base kernels. The method of [106] can be considered as a state-of-the-art boosting variant for this problem and operates on pairs of pixels. We use RBF kernels for BAHSIC as well while the other methods are linear.

Since each image can also be considered as a 252 dimensional feature vector a number of linear feature selection algorithms become applicable. So, we benchmark a number of other feature selection methods presented in the previous section besides two MKL formulations. Table 5.4 lists the feature selection results. Results are presented similarly to the table 5.1 Similar to UCI results, AdaBoost tended to perform the worst and selected only 12.6 features on average. The poor performance was due to the fact that each of the 252 SVMs was learnt independently. The weak classifier coefficients (i.e. kernel weights) did not influence the individual SVM parameters. By contrast, there is a very tight coupling between the two in MKL and GMKL and this ensures better performance. Of course, other forms of boosting do not have this limitation and the state-of-the-art boosting method of [106] performs better but is still significantly inferior to GMKL. Figure 5.2 shows the weight given to each pixel by the different feature selection algorithms (apart from the method of [106] which does not assign weights to individual pixels).

The performance of the linear feature selection methods is quite variable. The analysis is similar to that of UCI datasets except that OWL-QN performs poorly as the implicit bias weight got set to zero due to the l_1 regularization. On the other hand, BAHSIC, LP-SVM and Sparse-SVM perform even better than standard MKL (though not better than GMKL). The comparison between MKL and GMKL is even starker. GMKL achieves a classification accuracy of 93.2% using as few as 20 features and 95.1% using only 30 features. In comparison, standard MKL achieves just 83.8% and 86.3% respectively. This reinforces the observation that choosing the right kernel representation is much more important than converging to the globally optimal solution. Finally, the MKL and GMKL results for fixed uniform weights chosen via cross-validation are 92.6 ± 0.9 and 94.3 ± 0.1 respectively. Note that these results

Figure 5.1: Sample faces from the database of [10].



(a) Male (b) Selected (c) Female (d) Selected

(e) AdaBoost (f) OWL-QN (g) S-SVM (h) LP-SVM (i) Sum MKL (j) Prod MKL

Figure 5.2: A male and female face are shown in (a) and (c) while (b) and (d) depict the top 30 pixels selected using the Product MKL formulation. Classification accuracy using these 30 pixels is the same as that obtained using all the pixels. The pixel weights learnt on the first training testing split for the various feature selection methods are shown in (e)-(j) with black indicating small weights and white large weights. The number of pixels selected and classification accuracies are: (e) AdaBoost, 13 pixels and 76.07%; (f) OWL-QN [11] 51 pixels and 84.61%; (g) Sparse-SVM [12] 55 pixels and 91.31%; (h) LP-SVM [13] 50 pixels and 91.51%; (i) Sum MKL 146 pixels and 91.02%; and (j) Prod MKL 77 pixels and 96.43%.

are obtained using all 252 features. GMKL can obtain a similar classification accuracy using as few as 25 features.

In summary, there can be a difference of almost 10% between standard MKL and GMKL for a fixed small number of features and we can achieve an 8x compression ratio by sacrificing less than half a percent classification accuracy. This has many practical implications. For mobile and network applications that need to transmit images for classification, this can significantly prolong battery life and reduce bandwidth consumption. In other cases, this approach can be used to reduce the size of the training set. This can even enable the classifier to be loaded into the very limited RAM of the mobile or network camera. Finally in medical imaging, if an inexpensive preprocessing step can determine the alignment transformation, then our procedure can be used to significantly speed up the image acquisition and classification process.

Caltech 101

We now turn to the situation when the images are not well aligned. It is no longer meaningful to define kernels on individual pixels since a pixel lies on different parts of an object in different images. Since the objects are roughly aligned, images can instead be partitioned into a rectangular grid where there is alignment across grid cells (in a manner similar to a single level in spatial pyramid matching [114]). A kernel can now be defined over features computed in each grid cell and MKL can be applied to select kernels (grid cells).

While any feature, or sets of features, could be used we experiment with the Gist features of [115]. Images are partitioned into a grid of 8×8 cells and the mean filter responses calculated for each orientation and scale to get a 20 dimensional feature vector in each cell. An RBF kernel is defined over the feature vector in each cell to get 64 kernels. These are then combined using the standard MKL and GMKL formulation as in the previous case. Note that the linear feature selection methods tried out in the case of gender identification are no longer applicable. Boosting was applied in [116] to reduce the dimensionality of the Gist feature vector at the image level so as to preserve pairwise distances. However, both our methodology and objective are very different. Furthermore, we need to apply the multi-class MKL formulation in order to select a common set of image regions across all categories.

We test the GMKL formulation on the Caltech 101 database [14]. It consists of 102 classes of which 101 belongs to object categories and one is background class. Samples of different classes is shown in Figure 5.3. Even though the categories in the database show a lot of variability there are many classes which can be said to be roughly aligned in terms of scale, orientation and position. We employ the standard experimental setup [39] where we test on all 102 categories and use 15 images per category for training and a different 15 for testing.

We first look at some qualitative results to see which regions are getting selected in specific classes. This can reveal discriminative parts of objects as well as database artifacts and the role of context. For these qualitative results alone, the training set was split into disjoint training and validation sets. A 1-vs-All classifier was trained using the standard MKL formulation. This resulted in a short, ranked list of grid cells according to the learnt kernel weights. We further discarded those grid cells for which elimination did not result in a decrease in classification accuracy as measured on the validation set. The results are shown in Figures 5.4 and 5.5.

For the categories shown in Figure 5.4, mostly regions on the object are used to distinguish the class. For Windsor Chair, only 2 regions are selected. The back of the chair is quite distinctive but using that region alone causes confusion with a few other images which have a vertical striped texture in that region. Adding the region on the seat of the chair clears this ambiguity. For Motorbikes, the most distinguishing regions lie on the wheel. However, since the class is not perfectly aligned, a few Motorbike images have only their background visible in those regions. Adding the white border region which is present in most Motorbike images (an artifact of the database) helps classify some of these images but then other images with a white border start getting misclassified. Adding the region around the pillion takes care of such images. For Hedgehogs, the textured area on the back is the most important though the region around the eye is necessary to fully distinguish the category. For Faces Easy, the areas around the neck and face contour get the highest weight followed by the eye and hair. Dropping any of

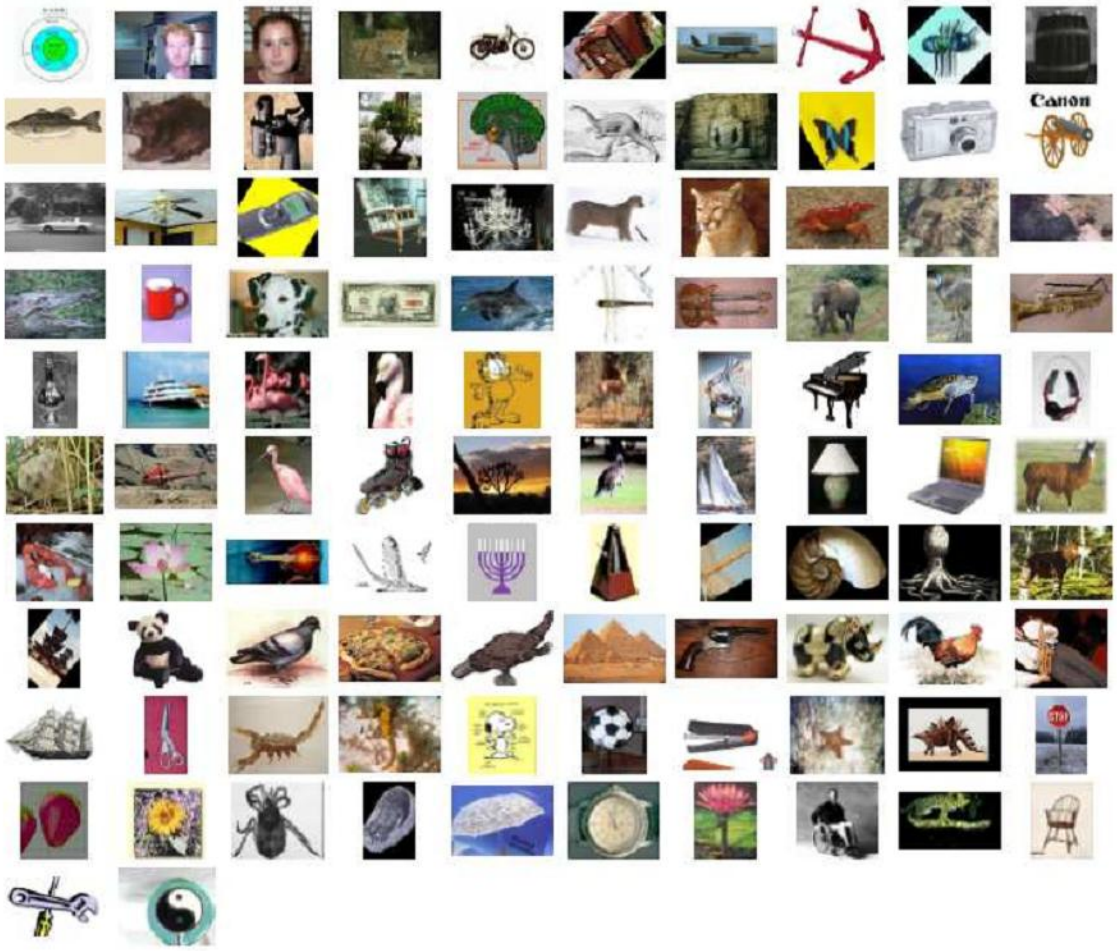


Figure 5.3: 101 classes of Caltech 101 [14] dataset.

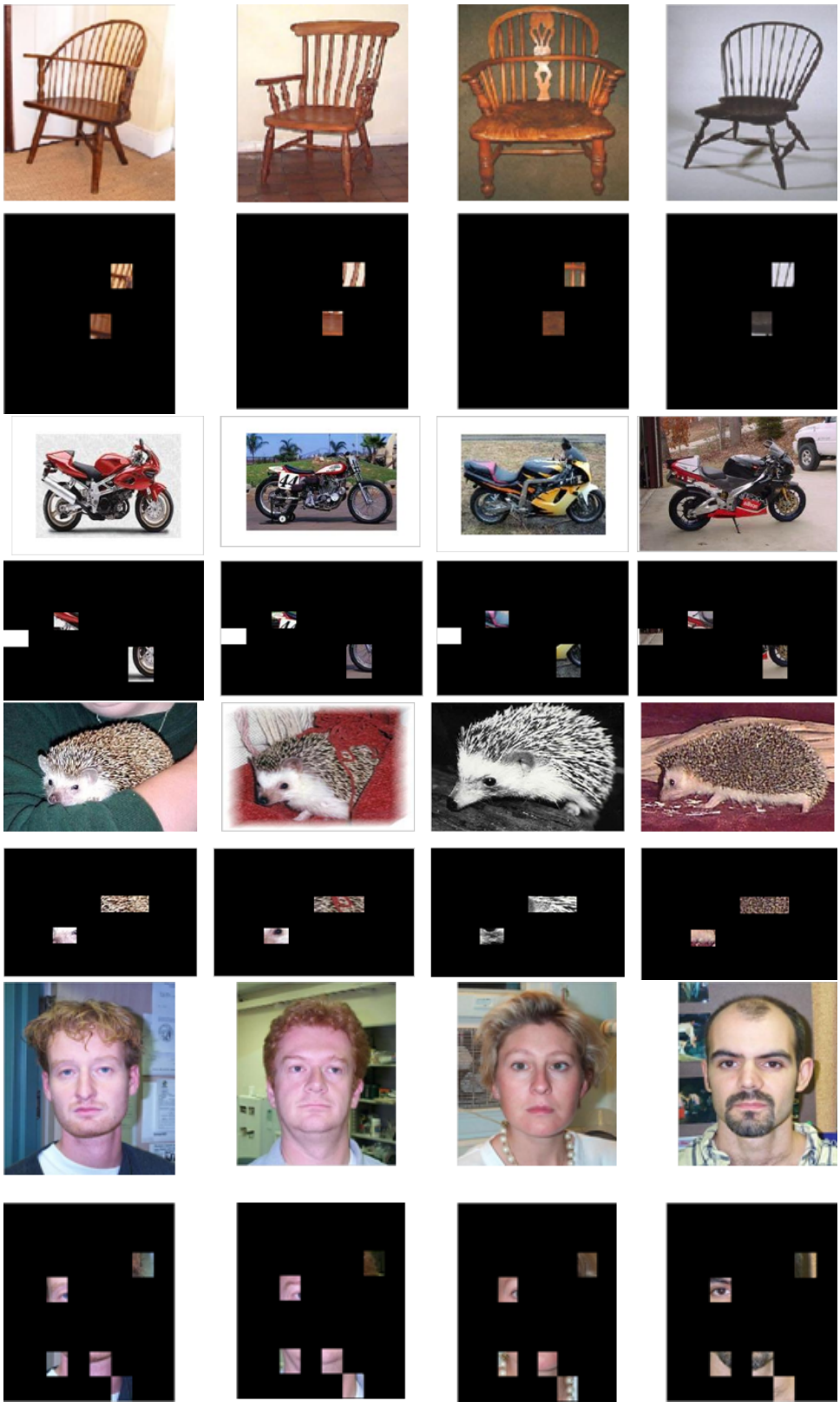


Figure 5.4: Sample images from the categories (from top to bottom) Windsor Chair, Motorbike, Hedgehog and Faces Easy and the regions selected by Sum MKL. Mostly regions on the object are being used for distinguishing the category.

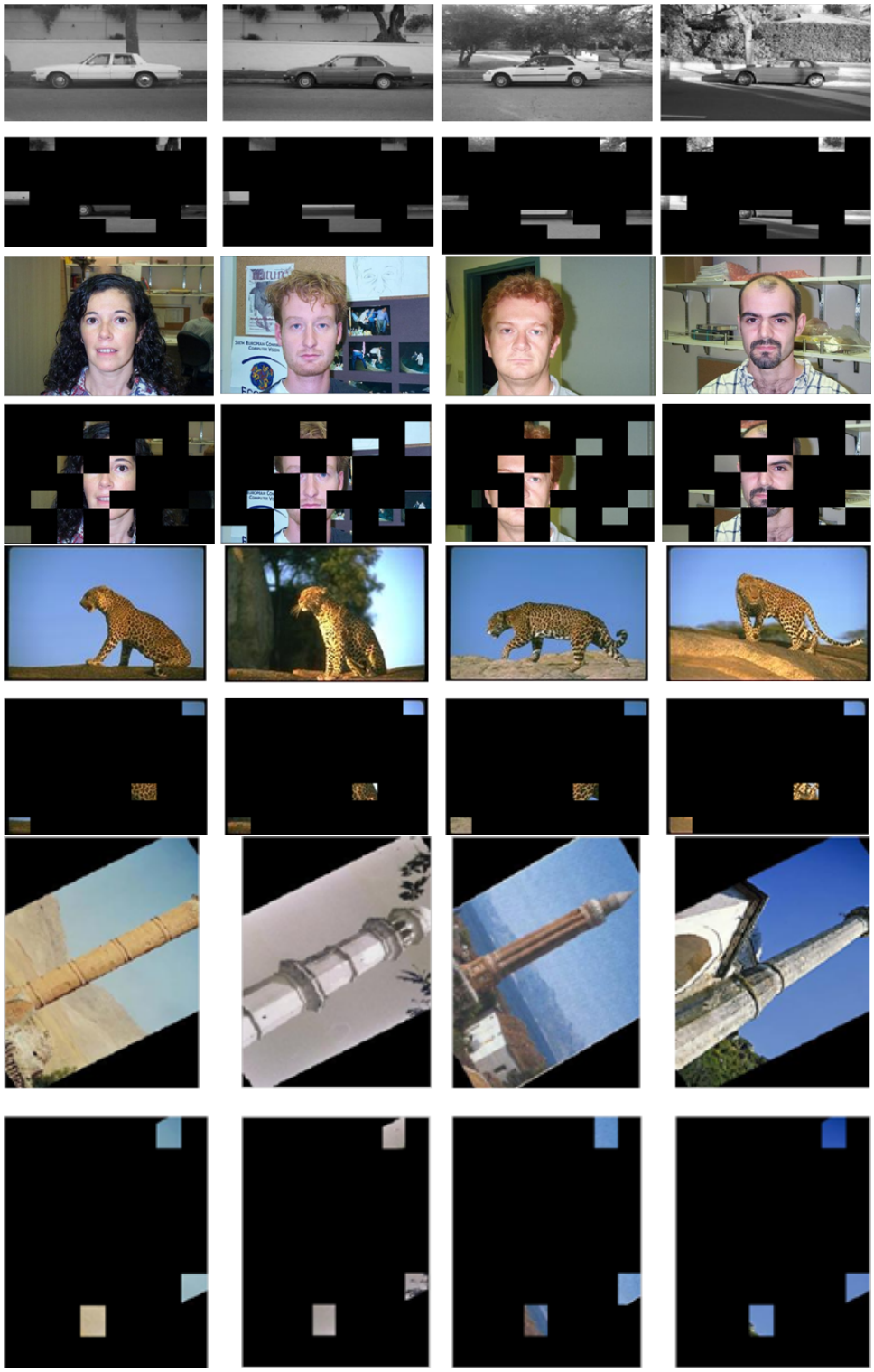


Figure 5.5: Sample images from the categories (from top to bottom) Car Side, Faces, Leopards and Minaret and the regions selected by Sum MKL. Mostly regions not lying on the object are used for distinguishing the category.

these regions causes misclassifications.

The categories shown in Figure 5.5 highlight the role of context and show artifacts of the database. The images in the category Car Side can vary a lot. The appearance of the road is much more stable and thus gets selected by the algorithm. However, to deal with the cases when the road is shadowed, some vegetation regions at the top of the image are also selected. The Faces category needs many regions since the images are not aligned and the algorithm needs to look at many regions to find the face. This also lets the algorithm look at the background which is necessary for distinguishing Faces from Faces Easy (which has exactly the same set of images but with the background cropped). For Leopards, the most distinguishing regions are at the image corners which are mostly sky and grass with a black border. The texture on the leopard is needed only to pin down a couple of images from other classes which also have a similar appearance in the corner regions. Finally, the images of minarets have all been pre-processed to have the same orientation. The rotation artifact provides the most important distinguishing region for this class. However, since other classes also have rotation artifacts as well as similar looking edges in that region, the algorithm chooses two extra sky regions to compensate. Some of these examples highlight that there are categories in the Caltech 101 database which can be recognized without even looking at the object of interest.

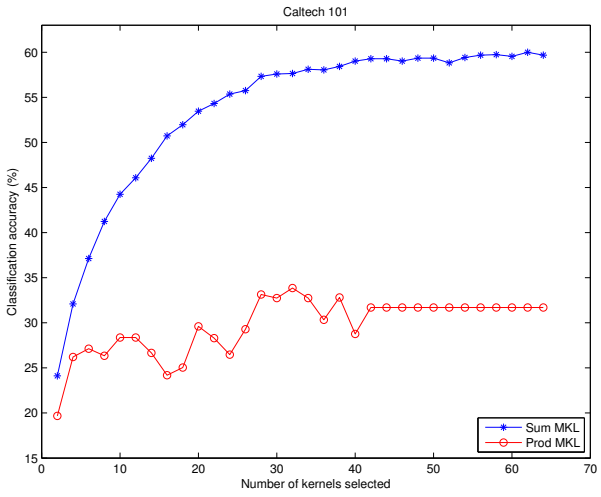


Figure 5.6: Variation in classification accuracy with number of image regions.

For a more quantitative measure, we apply the multi-class MKL formulation given in Section 4.4 to select a common set of image regions across all 102 classes. Using 1-vs-1 and 1-vs-all formulations a different set of kernel weights was learnt for each class. In the present context, this would imply that each binary 1-vs-1 or 1-vs-All classifier would select different set of features. When many classes are present this could result in examining whole set of features. Therefore we use other formulation where the kernel weights are jointly learnt with all the classifiers.

There are 64 kernel corresponding to the 64 grid cells and these are combined using the Sum and Product formulation. The results are plotted in Figure 5.6. Note that our intention is not to surpass the state-of-the-art in terms of classification accuracy by combining multiple features and getting even

	With Parameter Sharing		Without Parameter Sharing	
# Regions	MKL	GMKL	MKL	GMKL
17	61.00 (12)	65.00 (12)	59.00 (12)	58.00 (12)
34	68.00 (24)	69.00 (22)	64.00 (16)	65.00 (24)
54	69.00 (38)	79.00 (36)	66.00 (34)	68.00 (35)
79	72.00 (58)	85.00 (54)	70.00 (49)	71.00 (49)
102	72.00 (74)	87.00 (68)	70.00 (62)	74.00 (62)
172	72.00 (118)	88.00 (116)	73.00 (111)	74.00 (111)
300	72.00 (200)	88.00 (200)	73.00 (200)	74.00 (200)

Table 5.5: The variation in classification performance of Sum and Product MKL, with and without parameter sharing, as the number of selected regions is varied. The number of selected codewords is shown in brackets.

more complex representations. On the contrary, we aim to show that given any descriptor (or any set of descriptors) we can reduce the number of image regions with only a modest drop in classification performance. For instance, the classification accuracy obtained by standard MKL using all 64 kernels based on the Gist descriptor is 59.67% (this is comparable to the SVM based performance of leading individual features such as geometric blur (62.98%) and self similarity (60.83%) [39]). Our objective is to reduce the number of selected regions as much as possible while keeping classification accuracy above an acceptable threshold. As it turns out, virtually the same classification accuracy can be reached using only 40 regions. Using only 20 and 10 regions the classification accuracy drops to 90% and 75% of the original. The practical implications are similar to the case of gender identification in terms of increasing battery life and imaging speed or reducing bandwidth consumption. Note that the performance of GMKL is extremely poor in this case. It would appear that, in this case, the right feature representations is actually obtained by concatenating the individual feature maps rather then by taking their tensor product.

Caltech 256

In the previous two cases, where the images could be aligned, we could sample discriminative pixels and regions from a novel image without even looking at it. This is no longer possible if the object can occur at any orientation, scale and position in the image. Nevertheless, from a compression perspective, we can still use MKL to select a set of discriminative image regions. We tackle the problem by using MKL to select the most discriminative codewords in a codebook and by only keeping those image regions corresponding to the selected codewords. Note that we choose not to learn weights for features or image regions directly since we wish to avoid any training when determining regions for test images (since training after deployment is infeasible in most application scenarios).

In more detail, we use Geometric Blur [117] with a radius of 10 pixels to compute 300 features in each training image. The features are clustered using *K-Means* to learn 50 visual codewords per category. These are then aggregated across categories to form the codebook. While we have chosen a standard

way of generating the codebook, other methods could be used as well [118]. Given a codebook, image features are then labeled by the codewords via vector quantization to obtain a histogram or bag of visual words model.

To apply MKL, we associate an exponential χ^2 kernel with each codeword and combine them using the Sum and Product formulation. For Sum MKL, the kernels are combined as $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_l d_l e^{-\gamma_l \frac{(x_{il}-x_{jl})^2}{x_{il}+x_{jl}}}$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the histograms of image i and j respectively and the subscript k runs over the codebook entries (γ_l is set as in [39]). For Product MKL, we use the representation $k_{\mathbf{d}}(\mathbf{x}_i, \mathbf{x}_j) = \prod_l e^{-d_l \frac{(x_{il}-x_{jl})^2}{x_{il}+x_{jl}}}$. Note that both representations can be extended straight forwardly to the spatial pyramid matching case by defining the kernel for a particular codebook by summing over grid cells and pyramid levels. Many other methods have also been proposed for selecting codewords (for a by no means exhaustive list see [118–121] and references within) but these are methods based on mutual information, or information bottleneck or relative frequency of occurrence whereas we directly optimize for the classification task at hand. We also report results with and without multi-class parameter sharing. When the parameters are shared our method learns a common set of codewords across all classes. The without sharing case serves as a baseline. In this method we learn individual codeword rankings for each class using the 1-vs-All MKL formulation. The union of the top ranked codewords from each class is then taken to form the common set of selected codewords.

The proposed formulation is tested on the following 4 classes from the Caltech 256 [7] database: Elk, Fire Truck, French Horn and Teddy Bear. We choose 10 images from each class for training and 25 for testing. Since 50 codewords are learnt from each class we obtain a codebook of size 200 and therefore also have 200 kernels. Table 5.5 gives the results. In this case, GMKL with feature sharing does significantly better than MKL. This is understandable since products of kernels work in a much higher dimensional feature space and are more prone to over fitting. Parameter sharing helps overcome this problem and the results are much better than MKL with or without parameter sharing. For instance, using all 300 image regions GMKL gets 88% and this goes down by just 1% when using only 102 regions. By contrast, MKL gets only 73% and 72% respectively. Figure 5.7 shows some sample images and selected regions.

In terms of practical applications, our approach is most useful in mobile and network camera applications since they have enough computing power to extract features but typically not enough memory to store a training set. Our approach can be used to minimize the number of image regions transmitted back for classification or, if sufficient compression can be achieved, to compress training histogram models to fit in the device's memory.

5.4 Summary

We tackled the feature selection using GMKL. Proposed method gave good results on various datasets not only as compared to traditional MKL but also as compared to state-of-art wrapper and filter feature selection methods. We also tackled the problem of learning discriminative pixels and image regions for object categorization. Our objective is to reduce image representation size and we demonstrated that compression factors ranging from 1.5 to 8x can be achieved with less than a 1% drop in classification

Figure 5.7: Regions selected by GMKL in the Caltech 256 images.

accuracy. This has significant practical implications for object recognition based on mobile or network cameras.

We used our formulation to learn products of kernels and showed that this could improve performance by more than 10% in some cases. However, products of kernels are also more prone to over fitting and care should be taken to validate that they do lead to an appropriate feature representation for a given problem before they are applied.

Chapter 6

Character Recognition in Images

6.1 Introduction

With today's omnipresence of inexpensive portable convergent devices containing digital cameras and processors, the range of possible computer vision applications has experienced a fast growth. It is easy to envisage how PDAs will aid people who are currently unattended by technology. For instance, camera-phone with character recognition software will help the visually impaired to identify street signs, roads and shops names, grocery products, etc. This can also be of use for those who can not read text in the local language.

Automatic recognition of characters from images is a problem that has been approached since the early stages of computer vision, being an active research field since the mid 1950.s [122]. Under controlled situations this is one of the most successful applications of computer vision, and there are many solutions available commercially. In these cases, the image is usually acquired by scanning documents with a relatively high resolution, and usually a small amount of noise and distortion is allowed. Such methods are not expected to perform well for images obtained from photographs in which characters present perspective distortion, occlusion, variations in contrast, color, style and motion blur. However most of the research has focused characters from the Latin alphabet.

We work here towards automatic reading of text in natural scenes. In particular, our focus is on the *recognition* of individual characters in such scenes. Figures 6.2, 6.1 and 6.3 highlight why this can be a deceptively hard task. Even if the problems of clutter and text segmentation were to be ignored for the moment, the following sources of variability still need to be accounted for: (a) font style and thickness; (b) background as well as foreground colour and texture; (c) camera position which can introduce geometric distortions; (d) illumination and (e) image resolution. All these factors combine to give the problem a flavour of object recognition rather than optical character recognition or handwriting recognition. In fact, OCR techniques cannot be applied out of the box precisely due to these factors. Furthermore, viable OCR systems have been developed for only a few languages. and most Indic languages are still beyond the pale of current OCR techniques.

Many problems need to be solved in order to read text in natural images including text localization, word and character segmentation, recognition, integration of language models and context, *etc*. One can

Figure 6.1: Examples of high visual similarity between samples of different classes caused mainly by the lack of visual context.

approach the problem performing the above tasks in either top down or bottom up manner. Our focus, here, is on the basic character recognition aspect of the problem (see Figures 6.1, 6.2 , 6.5 and 6.6). We use database of images containing English and Kannada text [1] characters covering uppercase and lowercase of alphabets and numbers. In order to assess the feasibility of posing the problem as an object recognition task, we benchmark the performance of various features in a bag-of-visual-words (BoV) representation. Along we also use state of art classifier used for object recognition task, which is based on Multiple Kernel Learning (MKL). The results indicate that even the isolated character recognition task is challenging. The number of classes can be moderate (62 for English) or large (657 for Kannada) with very little inter-class variation as highlighted by Figures 6.1 and 6.2. This problem is particularly acute for Kannada where two characters in the alphabet can differ just by the placement of a single dot like structure. Furthermore, While training data is readily available for some characters others might occur very infrequently in natural scenes. We therefore investigate whether surrogate training data, either in the form of font generated characters or handwritten characters, can be used to bolster recognition in such a scenario. We also present baseline recognition results on the font and handwritten character databases to contrast the difference in performance when reading text in natural images.

6.2 Related Work

The task of character recognition in natural scenes is related to problems considered in camera-based document analysis and recognition. Most of the work in this field is based on locating and rectifying the text areas (e.g. [123], [124], [125] and [126]), followed by the application of OCR techniques [127]. Such approaches are therefore limited to scenarios where OCR works well. Furthermore, even the rectification step is not directly applicable to our problem, as it is based on the detection of printed document edges or assumes that the image is dominated by text.

Methods for off-line recognition of handwritten characters [128], [129] have successfully tackled the

[1] Available at http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/.

Figure 6.2: A small set of Kannada characters, all from different classes. Note that vowels often change a small portion of the characters, or add disconnected components to the character.

problem of intra-class variation due to differing writing styles. However, such approaches typically consider only a limited number of appearance classes, not dealing with variations in foreground/background colour and texture.

For natural scenes, some researchers have designed systems that integrate text detection, segmentation and recognition in a single framework to accommodate contextual relationships. For instance, [130] used insights from natural language processing and present a Markov chain framework for parsing images. [131] introduced composition machines for constructing probabilistic hierarchical image models which accommodate contextual relationships. This approach allows re-usability of parts among multiple entities and non-Markovian distributions. [132] proposed a method that fuses image features and language information (such as bi-grams and letter case) in a single model and integrates dissimilarity information between character images. The idea is that by comparing instances emitted by a source (e.g. characters from the same sign board), they help ensuring that similar instances are given the same label and vice-versa.

Simpler recognition pipelines based on classifying raw images have been widely explored for digits recognition (see [133], [134] and other works on the MNIST and USPS datasets). A more complex and robust approach is based on modelling this as a shape matching problem (e.g. [73]): several shape descriptors are detected and extracted and point-by-point matching is computed between pairs of images. In BoV-based methods, instead of matching feature vectors of pairs of images, each image is represented by unstructured feature counts. This robust representation is used as input to classifiers.

We use the BoV representation which, to the best of our knowledge, has not yet been applied for character recognition. We assess the suitability of six different feature extractors for our datasets. These feature extraction methods are state of art method which is widely used in computer vision for object recognition. For classification, we evaluate nearest neighbour, SVM and the multiple kernel learning method of [39], which has achieved state-of-the-art results on benchmark object recognition databases.

6.3 Datasets

Our focus is on recognizing characters in images of natural scenes. Towards this end, we compiled a database of English characters taken from images of street scenes. We also acquired a database of

hand-drawn characters and another of characters generated by computer fonts.

For English, we treat upper and lower case characters separately and include digits to get a total of 62 classes. Kannada does not differentiate between upper and lower case characters. It has 49 basic characters in its alphasyllabary, but consonants and vowels can combine to give more than 600 visually distinct classes.

6.3.1 Natural Images DataSet - *Img*

A set of 1922 images mostly of sign boards, hoardings and advertisements is collected. It also included a few images of products in supermarkets and shops. Some of these original images are shown in Figure 6.3.

Individual characters were manually segmented from these images. We experimented with two types of segmentations: rectangular bounding boxes and finer polygonal segments as shown in Figure 6.4. For the types of features investigated here, it turned out that polygonal segmentation masks presented almost no advantage over bounding boxes. Therefore, all the results presented in Section 6.5 are using the bounding box segmentations.

Our English dataset has 12503 characters, of which 4798 were labeled as bad images due to excessive occlusion, low resolution or noise. For our experiments, we used the remaining 7705 character images. Similarly, for Kannada, a total of 4194 characters were extracted out of which only 3345 were used. Figures 6.5 and 6.6 show examples of the extracted characters. These datasets will be referred to as the *Img* datasets.

6.3.2 Handwritten Dataset - *H*nd

The handwritten data set (*Hnd*) was captured using a tablet PC with the pen thickness set to match the average thickness found in hand painted information boards. For English, a total of 3410 characters were generated by 55 volunteers. For Kannada, a total of 16425 characters were generated by 25 volunteers. Some sample images are shown in Figure 6.7 ,6.8.

6.3.3 Font Dataset - *Fnt*

This dataset is obtained by synthesizing English characters using 254 different fonts in 4 styles (normal, bold, italic and bold+italic) to generate a total of 62992 (62 x 254 x 4) characters. Each images is of 128 x 128, for each character and font type, the font size is adjusted to generate a character that occupies most of the image; The images are in 256 grey levels, with white background and black foreground. The intermediate grey levels happen around edges as a result of antialiasing. This dataset will be referred to as the *Fnt* dataset.

Figure 6.3: Sample source images used to extract the characters for our data sets.1922 images were processed, of which, more than 17000 characters were extracted. 901 FrontalImages or 1352 Images + FrontalImages From FrontalImages, they extracted 12504 English characters and 5238 Kannada characters.

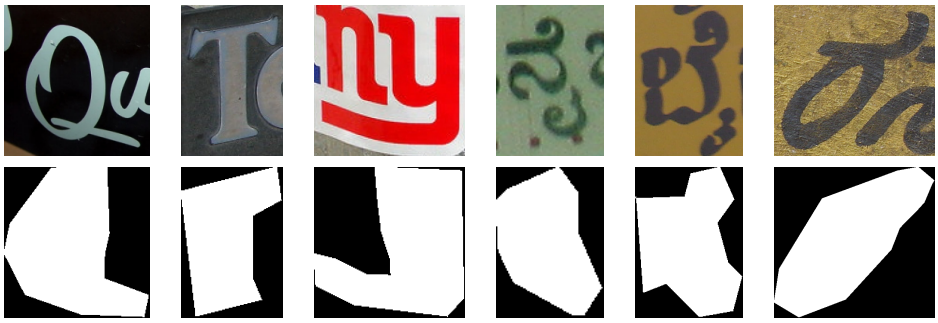


Figure 6.4: Sample characters and their segmentation masks.

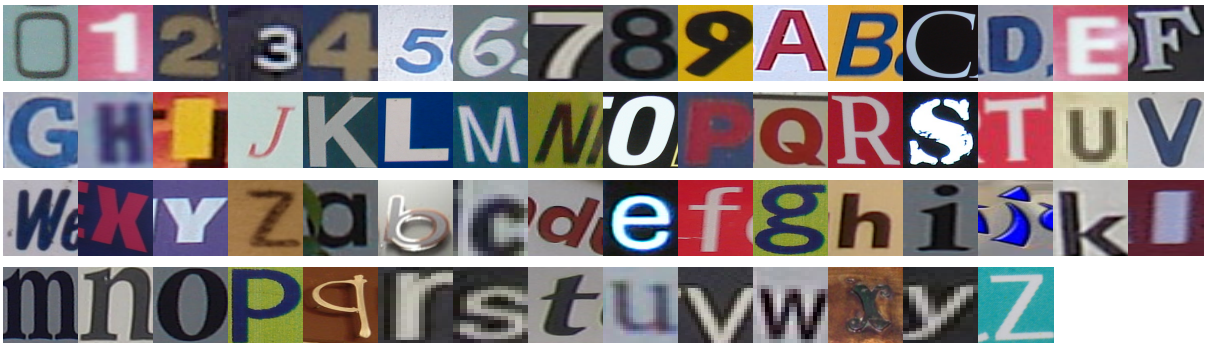


Figure 6.5: Sample characters of the *English images* set. 901 Frontal Images or 1352 Images + Frontal Images From Frontal Images, they extracted 12504 English characters and 5238 Kannada characters.

6.4 Feature Extraction and Representation

6.4.1 Bag of Words

Bag-of-visual-words is a popular technique for representing image content for object category recognition. The idea is to represent objects as histograms of feature counts. This representation quantizes the continuous high-dimensional space of image features to a manageable vocabulary of visual words. This is achieved, for instance, by grouping the low-level features collected from an image corpus into a specified number of clusters using an unsupervised algorithm such as K-Means (for other methods of generating the vocabulary see [135]). One can then map each feature extracted from an image onto its closest visual word and represent the image by a histogram over the vocabulary of visual words.

Most of the BoV-based works use unsupervised construction of the visual vocabularies. It is computationaly expensive to build such vocabularies, since samples from all classes are used. As an alternative, we chose the method of [76], which builds one visual vocabulary per class and combines the resulting representations by concatenation of the vectors. This may lead to richer representations, since the class labels are taken into account. For all the feature extraction methods, we used a dictionary that consists of 5 centroids per class. Thus, for the English characters database, each sample is described by a histogram of 310 dimensions. The dictionaries were built using K-means with the χ^2 statistics to evaluate the distance between feature vectors.

Figure 6.6: A random selection of the Kannada images database.

Figure 6.7: Samples hand-drawn characters of the English data sets.

6.4.2 Feature Extraction

We evaluated six different types of local features. Not only did we try out shape and edge based features, such as Shape Context, Geometric Blur and SIFT, but also features used for represent- ing texture, such as filter responses, patches and Spin Images, since these were found to work well in [136]. We explored the most commonly used parameters and feature detection methods employed for each descriptor, with a little tuning, as described below.

As a pre-processing step, the images are normalize for zero mean and unit variance over the grey level values. The list below details the parameters chosen for these descriptors. These choices were based on a number of preliminary experiments.

- **Shape Contexts** (SC) [73] is a descriptor for point sets and binary images. We sample points using the Sobel edge detector. The descriptor is a log-polar histogram, which gives a $\theta \times n$ vector, where θ is the angular resolution and n is the radial resolution. We used $\theta = 15$ and $r = 4$,

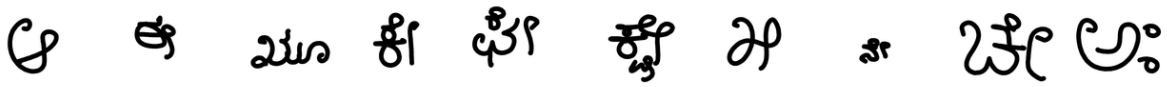


Figure 6.8: Samples hand-drawn characters of the Kannada data sets.

inner radius (r_i) is set to $\frac{1}{8}$ and outer radius (r_o) is set to 1. The above parameters result in SC descriptors of dimension 60. These parameters present a good trade-off between radial resolution and angular resolution. No rotation invariance was used.

- **Geometric Blur** (GB) [74] is a feature extractor with a sampling method similar to that of SC, but instead of histogramming points, the region around an interest point is blurred according to the distance from this point. For each region, the edge orientations are counted with a different blur factor. This avoids quantisation problems of SC and allows its application to grey scale images.

- **Scale Invariant Feature Transform** (SIFT) [72] are extracted on points located by the Harris Hessian-Laplace detector, which gives affine transform parameters. The feature descriptor is computed as a set of orientation histograms on (4×4) pixel neighborhoods. The orientation histograms are relative to the keypoint orientation. The contribution of each pixel is weighted by the gradient magnitude, and by a Gaussian with σ 1.5 times the scale of the keypoint. The histograms contain 8 bins each, and each descriptor contains a 4×4 array of 16 histograms around the keypoint. This leads to feature vector with 128 elements. In our experiments, the detector usually located at least 100 keypoints per image, but for some images not enough points were obtained.

- **Spin image** [75], [137] is a two-dimensional histogram encoding the distribution of image brightness values in the neighborhood of a particular reference point. The two dimensions of the histogram are d, distance from the centre point, and i, the intensity value. Since d and i are invariant under orthogonal transformations of the image neighborhood, Spin images are invariant to affine-normalised patches. We used 11 bins for distance and 5 for intensity value, resulting in 55-dimensional descriptors. The same interest point locations used for SIFT were used for spin images.

- **Maximum Response of filters** (MR8) [76] is a texture descriptor based on a set of 38 filters but only 8 responses. The filters include a Gaussian and a Laplacian of a Gaussian (LOG) filter both at scale $\theta = 10$, an edge (first derivative) filter at 6 orientations and 3 scales and a bar (second derivative) filter at 6 orientations and the same 3 scales $(\theta_x, \theta_y) = \{(1, 3), (2, 6), (4, 12)\}$. The response of the isotropic filters (Gaussian and LOG) are used directly, but the responses of the oriented filters (bar and edge) are collapsed at each scale by using only the maximum filter responses across all orientations, there by ensuring rotation invariance. This filter is extracted densely, (sampled at each 25×25 patch), giving a large set of 8D vectors. Weber normalization is applied to the individual feature vectors and a threshold disregards continuous regions.

- **Patch descriptor** (PCH) is the simplest dense feature extraction method. For each position, the raw $n \times n$ pixel values are vectorised, generating an n^2 descriptor. Weber normalization is applied to each vector individually. We used 5×5 patches. With a small threshold on the standard deviation of the pixel values to disregard uniform areas. Images are scaled to 128×128. For each patch position, the 5×5 pixels are extracted from a 20×20 pixels area by sub-sampling one pixel for each four pixels per row and column.

6.5 Experiments

This section describes baseline experiments with three classification schemes: (a) nearest neighbor classification using χ^2 statistics as a metric; (b) support vector machines (SVM); and (c) multiple kernel learning (MKL). Additionally, on English datasets we show results obtained by the commercial OCR system ABBYY FineReader 8.0. For an additional benchmark, we provide results obtained with the dataset of the ICDAR Robust Reading competition 2003[2]. This set contains 11615 images of characters used in English. The images are more challenging than our English *Img* dataset and it has some limitations, such as the fact that only few samples are available for some of the characters.

Most of our experiments were done with our *English Img* characters dataset. It is demonstrated that the performance of MKL using only 15 training images is nearly 25% better thatn that of ABBYY FineReader, a commercial OCR system. Also, when classifying the *Img* test set, if appropriate features such as Geometeri Blur, are used, then a NN classifier trained on the synthetic *Fnt* dataset is a as good as the NN classiifer trained on an equal number of *Img* samples. Futher more, since synthetic *Fnt* data is easy to generate, an NN classifier trained on a large *Fnt* data is easy to generate, an NN classifier trained on a large *Fnt* training set can perform nearly as well as MKL trained on 15 *Img* samples per class. This opens up the possibility of improving classification accuracy without having to acquire expensive *Img* training data.

6.5.1 English Datasets

Homogeneous Sets

The six feature extraction methods were evaluated for the three data sets. Here we show results obtained by training and testing with samples from the same type: *Fnt*, *Hnd* and *Img*. For some classes, the number of available samples of natural images was just above 30, so we chose to keep the experiment sets balanced and fix the test set size to 15 samples per class for all the three databases. For the test sets, we varied the number of training samples. This was repeated with random selections of training samples. The number of samples available for training (with no intersection with the test set) was 1001, 40 and 15 for *Fnt*, *Hnd* and *Img*, respectively. The number of training splits selected at random and results are averaged. Table 6.1 shows the results obtained with training sets of 15 samples per class.

The performance of GB and SC is significantly better than all the other features. Also, there can be more than a 20% drop in perfomance when moving from training and testing on *Fnt* or *Hnd* to training and testing on *Img*. This indicates how much more difficult recognizing characters in natural images can be.

The features were also evaluated using SVM with RBF kernel for the *Img* dataset, leading to the results shown in table 6.2. The kernel paramter gamma is choosen through cross-validation. As expected, SVM lead to an increase in performance with most features, except with Patches. The evaluated implementation was a multi-class SVM with one-vs-all classification scheme.

An additional experiment was performed with the multiple kernel learning method of [39], which gave state-of-the-art results in the Caltech256 challenge. We combined all the six feature extraction

[2] http://algoval.essex.ac.uk/icdar

Feature	Fonts	Handwritten	Images
GB	69.71 ± 0.64	65.40 ± 0.58	47.09
SC	64.83 ± 0.60	67.57 ± 1.40	34.41
SIFT	46.94 ± 0.71	44.16 ± 0.79	20.75
Patches	44.93 ± 0.65	69.41 ± 0.72	21.40
SPIN	28.75 ± 0.76	26.32 ± 0.42	11.83
MR8	30.71 ± 0.67	25.33 ± 0.63	10.43
ABBYY	66.05 ± 0.00	–	30.77
# train splits	10	5	1

Table 6.1: Nearest neighbour classification results (%) obtained by different feature extractors on the English data sets. These were obtained with 15 training and 15 testing samples per character class chosen. For comparison, the results with the commercial software ABBYY are also shown. The bottom row indicates how many sets of training samples were taken per class to estimate mean and standard deviation of the classification results.

GB	52.58
SC	35.48
SIFT	21.40
Patches	21.29
SPIN	13.66
MR8	11.18
MKL	55.26

Table 6.2: Classification results (%) obtained with SVM and with MKL (combining all the features) for the *Img* set with 15 training samples per class.

methods at kernel level and performed classification experiments using the one-vs-all scheme. Using such classifiers ensure the optimal combinations of different aspects of data optimally. This resulted on the accuracy of 55.26% using 15 training samples per class. This represents an improvement of less than 3% over the result of the best performing feature alone (Geometric Blur).

As can be seen from these experiments, it is possible to surpass the performance of ABBYY, a state-of-the-art commercial OCR system, using 15 training images even on the synthetic *Fnt* dataset. For the more difficult *Img* dataset the difference in performance between MKL and ABBYY is nearly 25% indicating that OCR is not suitable for this task. Nevertheless, given that the performance using MKL is only 55%, there is till tremendous scope for improvement in the object recognition framework.

We also performed experiments with the ICDAR dataset, obtaining the results in Table 6.3. Due to the limitations of this dataset, we fixed the training set size of 5 samples per class and evaluated it in comparison to our dataset. As can be seen, the ICDAR results are worse than the *Img* results indicating that this might be an even tougher database. If we train on *Img* and test on ICDAR then the result can improve as more training data is added (see Table 6.4).

Ftr.	Images	ICDAR
GB	36.9 ± 1.0	27.81
SC	26.1 ± 1.6	18.32
PCH	13.7 ± 1.4	9.67
MR8	6.9 ± 0.7	5.48

Table 6.3: Nearest neighbour results obtained with 5 training samples per class for some of the features. Here we compare our English *Img* dataset and with the ICDAR dataset.

Tr. Spls.	15/class	all
GB	32.72	40.97
SC	27.90	34.51

Table 6.4: Nearest neighbour results obtained by training with English *Img* and testing with the ICDAR dataset – using 15 training samples per class and using the whole *Img* set for training.

Hybrid Sets

In this subsection we show experiments with hybrid sets, where we train on data from the *Fnt* and *Hnd* datasets and test on the same 15 images per class from the *Img* test set used in the previous experiments. The results are shown in Table 6.5 and indicate that for features such as Geometeric Blur, training on easily available syntheitc fonts is as good as training on original *Img* data. However, the performance obtained by training on *Hnd* is poor.

Feature	Training on *Fnt*	Training on *Hnd*
GB	47.16 ± 0.82	22.95 ± 0.64
SC	32.39 ± 1.39	26.82 ± 1.67
SIFT	9.86 ± 0.91	4.02 ± 0.52
Patches	5.65 ± 0.69	1.83 ± 0.44
SPIN	2.88 ± 0.68	2.71 ± 0.33
MR8	1.87 ± 0.60	1.61 ± 0.11
# test splits	10	5

Table 6.5: Nearest neighbour results with mixed data type: testing the recognition of natural images using training data from fonts and handwritten sets, both with 15 training samples per class. These results should be compared with the *Img* column of Table 6.1.

To aid visualization of the results, Figure 6.9 shows results of the experiments described above, separating panels for the top three methods: Geometric Blur, Shape Contexts and Patches. There is one curve for each type of experiment, where *FntImg* indicates training with *Fnt* and testing with *Img*, and *HndImg* indicates training with *Hnd* and testing with *Img*. The other curves show results by training and testing with the same kind of set (*Fnt*, *Hnd* and *Img*). Note that, for Geometric Blur, the NN performance

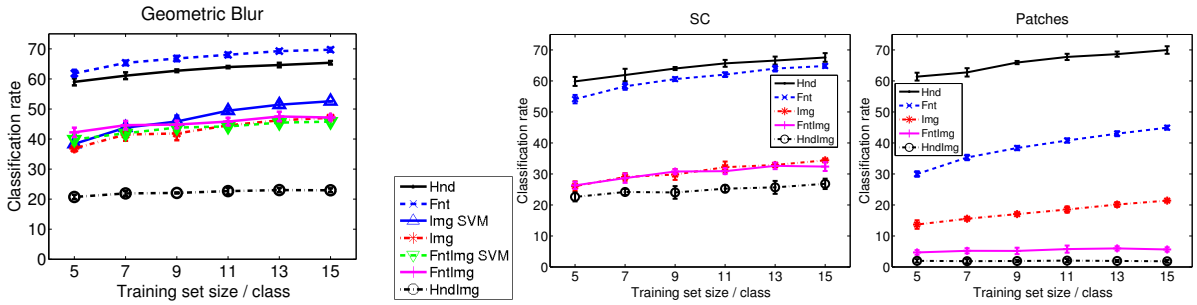


Figure 6.9: Classification results for the English datasets with the top two feature extraction methods: Geometric Blur (left), Shape Contexts (centre) and Patches (right). The plots show the mean and STD (error bars) varying with the size of the training sets. These were taken as sub-sets of the 15-samples-per-class sets of tables 6.1 and 6.5.

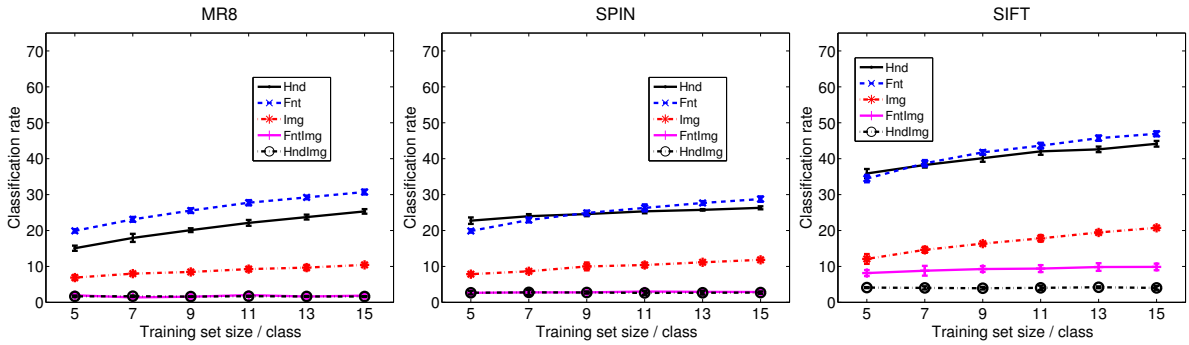


Figure 6.10: Classification results for the English datasets with the other feature extraction methods: MR8 filter banks(left), SPIN (centre) and SIFT (right). The plots show the mean and STD (error bars) varying with the size of the training sets. These were taken as sub-sets of the 15-samples-per-class sets of tables 6.1 and 6.5.

when trained on *Fnt* and tested on *Img* is actually better than NN performance when trained and tested on *Img*

In a practical situation, all the available fonts or hand-printed data could be used to classify images. Table 6.6 shows the results obtained by training with all available samples from *Fnt* and *Hnd* and testing with the same test sets of 15 samples per class described above. Note that for GB and SC, the results obtained by training with *Fnt* were better than those obtained by training with *Img* shown in table 6.1. This demonstrates the generalisation power of these descriptors and validates the possibility of cheaply generated large sized synthetic sets and using them for training.

Figure 6.11 shows the confusion matrix obtained for MKL when trained and tested on 15 *Img* samples per class.One can notice two patterns of high values in parallel to the diagonal line. These patterns show that, for many characters, there is a confusion between lower case and upper case. If we classify characters in a case insensitive manner, the accuracy turns out to be 57.20% (a 10% increase) for GB on *Img* and 80.80% (a 11% increase) for GB on *Fnt*, both using 15 training samples per class.

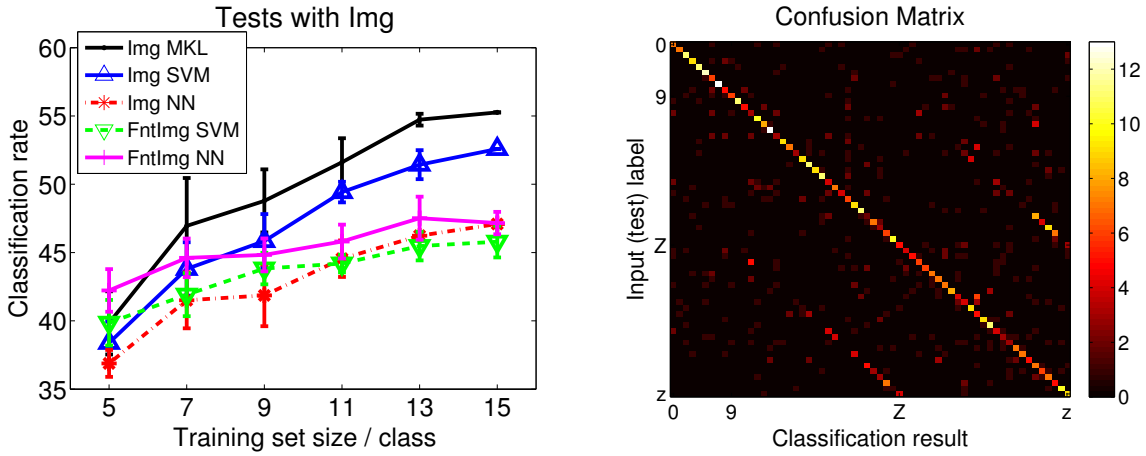


Figure 6.11: **Top**: results with the multiple kernel combination of all the features (MKL, solid black line), training and testing with *English Img*. For comparison, this panel also shows the results with the best individual feature, Geometric Blur (as shown in Figure 6.9-left). **Bottom**: the confusion matrix of MKL for this experiment with 15 training samples per class.

6.5.2 Kannada Data Sets

The *Img* dataset of Kannada characters was annotated per symbol, which includes characters and syllable, resulting in a set of 990 classes. Since some of these classes occur rarely in our dataset, we did not perform experiments training and testing with *Img*. Instead, we only performed experiments on training with *Hnd* characters and testing with *Img*. We selected a subset of 657 classes which coincides with the classes acquired for the *Hnd* dataset.

Table 6.7 shows baseline results. For these experiments, random guess would have a 0.15% accuracy. The low accuracies show how difficult are indic languages , when compared to English.

Feature	Training on *Fnt*	Training on *Hnd*
GB	54.30	24.62
SC	44.84	31.08
SIFT	11.08	3.12
Patches	7.85	1.72
SPIN	3.44	2.47
MR8	1.94	1.51
Training set size	1016	55

Table 6.6: Classification results (%) obtained with the same testing set as in table 6.5, but here the whole sets of synthetic fonts and handwritten characters are used for training, i.e., 1016 and 55 samples per class, respectively.

Ftr.	Trn/tst on *Hnd*	Trn on *Hnd*, tst on *Img*
GB	17.74	2.77
SC	29.88	3.49
SIFT	7.63	0.30
Patches	22.98	0.12
SPIN	2.37	0.16
MR8	5.12	0.00

Table 6.7: Nearest neighbour results (%) for the Kannada datasets: (i) training with 12 *Hnd* and testing with 13 *Hnd* samples, and (ii) training with all *Hnd* and testing with all *Img* samples.

6.6 Summary

In this Chapter, we tackled the problem of recognizing characters in images of natural scenes. We use a database of images of street scenes taken in Bangalore, India and showed that even commercial OCR systems are not well suited for reading text in such images. Working in an object categorization framework, we were able to improve character recognition accuracy by 25% over an OCR based system. The best result on the English Img database was 55.26% and was obtained by the multiple kernel learning (MKL) method of [39] when trained using 15 Img samples per class. This could be improved further if we were not to be case sensitive. Nevertheless, significant improvements need to be made before an acceptable performance level can be reached.

Obtaining and annotating natural images for training purposes can be expensive and time consuming. We therefore explored the possibility of training on hand-printed and synthetically generated font data. The results obtained by training on hand-printed characters were not encouraging. This could be due to the limited variability amongst the writing styles that we were able to capture as well as the relatively small size of the training set. On the other hand, using synthetically generated fonts, the performance of nearest neighbor classification based on Geometric Blur features was extremely good. For equivalent size training sets, training on fonts using a NN classifier could actually be better than training on the natural images themselves. The performance obtained when training on all the font data was nearly as good as that obtained using MKL when trained on 15 natural image samples per class. This opens up the possibility of harvesting synthetically generated data and using it for training.

As regards features, the shape based features, Geometric Blur and Shape Context, consistently outperformed SIFT as well as the appearance based features. This is not surprising since the appearance of a character in natural images can vary a lot but the shape remains somewhat consistent.

We also presented preliminary results on recognizing Kannada characters but the problem appears to be extremely challenging and could perhaps benefit from a compositional or hierarchical approach given the large number of visually distinct classes.

Chapter 7

Conclusions & Future work

7.1 Summary and Contributions

We have shown how MKL formulations can be generalized to learn general kernel combinations subject to general regularization on the kernel parameters. While our focus was on binary classification our approach can be applied to other loss functions and even other formulations such as Local MKL, multi-class and multi-label MKL. Generalized kernel learning can be achieved very efficiently based on gradient descent optimization and existing large scale SVM solvers. As such, it is now possible to learn much richer feature representations as compared to standard MKL without sacrificing any efficiency in terms of speed of optimization.

Proposed GMKL formulation based on products of kernels was shown to give good results for various feature selection problems – not only as compared to traditional MKL but also as compared to leading wrapper and filter feature selection methods. Of course, taking products of kernels might not always be the right approach to every problem. In such cases, our formulation can be used to learn other appropriate representation including sums of kernels. Finally, it should be noted that the classification accuracy of GMKL with learnt weights tends to be much the same as that obtained using uniform weights chosen through cross-validation. The advantage in learning would therefore seem to lie in the fact that GMKL can learn to achieve the same classification accuracy but using far fewer features. We also tackled the problem of learning discriminative pixels and image regions for object categorization. Our objective is to reduce image representation size and we demonstrated that compression factors ranging from 1.5 to 8x can be achieved with less than a 1% drop in classification accuracy. This has significant practical implications for object recognition based on mobile or network cameras. We tackled the problem of recognizing characters in images of natural scenes. We show that even commercial OCR systems are not well suited for reading text such images. Working in an object categorization framework, we were able to improve character recognition accuracy by 25% over an OCR based system using MKL.

In summary, we proposed GMKL which can learn non-linear kernel combinations. Shown how effective it is, in tackling feature selection and learning discriminative parts for object categorization. Shown how the MKL can improve the performance on the problem of recognition of character images taken in natural scenes. The applications considered in the thesis are of practical importance, and results

demonstrate that learning much richer combinations of kernels can represent data in more appropriate manner.

7.2 Future Scope

Although, MKL has become powerful tools for data analysis, finding the appropriate representation is still an active research area. More experiments and applications towards regression can be done. Possible extensions of the work includes (i) scaling to large kernel matrix (ii) extending it to other methods of multi-class classification (iii) solving multi-label multi-class problems. (iv) scaling to larger number of kernels using some parallel computing techniques.

Not just limiting to classification or regression problems this techniques can be explored towards dimensionality reduction or clustering algorithms. MKL can also be explored in specific to applications in different domains like in computer vision to the problems of object detection, visual learning, etc.

Related Publications

- M. Varma and B. Rakesh Babu, "More Generality in Efficient Multiple Kernel Learning", in *Proceedings of the International Conference on Machine Learning*, Montreal, Canada, 2009.
- T. E. de Campos, B. Rakesh Babu and Manik Varma, "Character recognition in natural images", in *Proceedings of the International Conference on Computer Vision Theory and Applications*, Lisbon, Portugal, 2009.

Appendix

Appendix A

Notation

Symbol	Description
SVM	Support Vector Machine
MKL	Multiple Kernel Learning
GMKL	Generalized multiple kernel learning
$\mathbf{x}, \mathbf{y}$	Input samples in vector form of $m-$ dimensional
$k(\mathbf{x}, \mathbf{y})$	Kernel Function
$\mathbf{K}$	Kernel Matrix
M	Number of training samples
N	Number of classes
K	Number of kernels
V	Vocabulary size
$\mathbf{a}^t$	Transpose of vector $\mathbf{a}$
$\langle , \rangle$	Inner product

Appendix B

Derivation of MKL

To find the best coefficients for the linear combination kernel in MKL $k_{opt} = \sum_l d_l k_l$, following objective function can be used. Minimize

$$J = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i^M \xi_i + \sum_l^K d_l\sigma_l \tag{B.1}$$

$$\text{subject to} \quad y_i(\mathbf{w}^t\Phi(\mathbf{x_i}) + b) \geq 1 - \xi_i \quad \forall i \tag{B.2}$$

$$\xi_i > 0,\ y_i \in \{-1, 1\} \quad \forall i \tag{B.3}$$

$$\mathbf{d} \geq 0, \mathbf{Ad} \geq \mathbf{p} \tag{B.4}$$

$$\Phi(\mathbf{x_i})^t\Phi(\mathbf{x_j}) = \sum_{l=1}^K d_l\Phi(\mathbf{x_i})_l^t\Phi(\mathbf{x_j})_l \tag{B.5}$$

where $\mathbf{d}$ are kernel parameters and $\mathbf{A}, \mathbf{p}$ are the parameters to include prior knowledge on kernel parameter $\mathbf{d}$.

$$Q(\mathbf{w}, b, \xi, \mathbf{d}, \alpha, \beta, \gamma, \delta) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^M \xi_i + \sum_{l=1}^K d_l\sigma_l - \sum_{i=1}^M \alpha_i\left[y_i(\mathbf{w^T\Phi}(\mathbf{x_i}) + b) - 1 + \xi_i\right] - \sum_{i=1}^M \beta_i\xi_i - \sum_{l=1}^K \gamma_l d_l - \sum_{l=1}^K d_l\boldsymbol{\delta}^t\mathbf{A_l} + \mathbf{p^t}\boldsymbol{\delta} \tag{B.6}$$

[1]

where $\alpha_i, \beta_i, \gamma_l, \delta_l$ are the non-negative Lagrangian multipliers and $\mathbf{A_l}$ is lth column of the matrix $\mathbf{A}$.

[1] Note that $\boldsymbol{\delta^t}(\mathbf{Ad} - \mathbf{p})$ is rewritten as $\sum_{l=1}^K d_l\boldsymbol{\delta}^t\mathbf{A_l} - \mathbf{p^t}\boldsymbol{\delta}$

For the optimal solution following KKT conditions are satisfied,

$$\frac{\partial Q}{\partial \mathbf{w}} = \mathbf{0} \Longrightarrow \mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \Phi(\mathbf{x_i}) \tag{B.7}$$

$$\frac{\partial Q}{\partial b} = \mathbf{0} \Longrightarrow \sum_{i=1}^{M} \alpha_i y_i = 0 \tag{B.8}$$

$$\frac{\partial Q}{\partial \xi_i} = \mathbf{0} \Longrightarrow C = \alpha_i + \beta_i \tag{B.9}$$

$$\frac{\partial Q}{\partial d_l} = 0 \implies -\sum_{i=1}^{M} \alpha_i y_i (\mathbf{w}^t \frac{\partial \Phi(\mathbf{x_i})}{\partial d_l}) + \sigma_l - \gamma_l - \boldsymbol{\delta}^t \mathbf{A}_l = 0$$

$$-\sum_{i=1}^{M} \alpha_i y_i (\mathbf{w}^t [\mathbf{0}_1 \cdots \Phi_l(\mathbf{x_i}) \cdots \mathbf{0}_K]^t \frac{1}{2\sqrt{d_l}}) + \sigma_l - \gamma_l - \boldsymbol{\delta}^t \mathbf{A}_l = 0$$

$$(\text{ since } \quad \Phi(\mathbf{x_i}) = [\sqrt{d_1}\Phi_1(\mathbf{x_i}) \ldots \sqrt{d_l}\Phi_l(\mathbf{x_i}) \ldots . \sqrt{d_K}\Phi_K(\mathbf{x_i})]^t)$$

$$\sigma_l = \gamma_l + \boldsymbol{\delta}^t \mathbf{A_l} + \sum_{i=1}^{M} \alpha_i y_i (\mathbf{w}^t [\mathbf{0}_1 \cdots \Phi_l(\mathbf{x_i}) \cdots \mathbf{0}_K]^t) \frac{1}{2\sqrt{d_l}}$$

Substituting Equation B.7

$$\sigma_l = \gamma_l + \boldsymbol{\delta}^t \mathbf{A_l} + \sum_{i=1}^{M} \alpha_i y_i ((\sum_{i=1}^{M} \alpha_i y_i \Phi(\mathbf{x_i}))^t [\mathbf{0}_1 \cdots \Phi_l(\mathbf{x_i}) \cdots \mathbf{0}_K]^t) \frac{1}{2\sqrt{d_l}}$$

$$\sigma_l = \gamma_l + \boldsymbol{\delta}^t \mathbf{A_l} + \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j [\sqrt{d_1}\Phi_1(\mathbf{x_i}) \ldots \sqrt{d_l}\Phi_l(\mathbf{x_i}) \ldots . \sqrt{d_K}\Phi_K(\mathbf{x_i})]^t [\mathbf{0}_1 \cdots \Phi_l(\mathbf{x_i}) \cdots \mathbf{0}_K]^t) \frac{1}{2\sqrt{d_l}}$$

$$\sigma_l = \gamma_l + \boldsymbol{\delta}^t \mathbf{A_l} + \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j \Phi_l(\mathbf{x_i})^t \Phi_l(\mathbf{x_i}) \tag{B.10}$$

$$\alpha_i \{ y_i (\mathbf{w}^t \mathbf{x_i} + b) - 1 + \xi_i \} = 0 \quad \forall\, i \tag{B.11}$$

$$\beta_i \xi_i = 0 \quad \forall\, i \tag{B.12}$$

$$\alpha_i \geq 0, \beta_i \geq 0, \xi_i \geq 0, d_l \geq 0 \quad \forall\, i, l \tag{B.13}$$

$$\gamma_l d_l = 0 \quad \forall\, l \tag{B.14}$$

$$\boldsymbol{\delta^t}(\mathbf{Ad} - \mathbf{p}) = \mathbf{0} \tag{B.15}$$

where $\mathbf{0}_l$ is a vector containing all zeros of size $\Phi_l(\mathbf{x_i})$.

Taking Equation (B.6), i.e.,

$$Q = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{M}\xi_i + \sum_{l=1}^{K} d_l\sigma_l - \sum_{i=1}^{M}\alpha_i\left[y_i(\mathbf{w^T\Phi(x_i)} + b) - 1 + \xi_i\right] - \sum_{i=1}^{M}\beta_i\xi_i - \sum_{l=1}^{K}\gamma_l d_l - \sum_{l=1}^{K} d_l\boldsymbol{\delta}^t\mathbf{A_l} + \mathbf{p^t}\boldsymbol{\delta}$$

and by substituting equation (B.7)

$$= \frac{1}{2}(\sum_{i=1}^{M}\alpha_i y_i\Phi(\mathbf{x_i}))^t(\sum_{j=1}^{M}\alpha_j y_j\Phi(\mathbf{x_j})) - \sum_{i=1}^{M}\alpha_i(y_i((\sum_{j=1}^{M}\alpha_j y_j\Phi(\mathbf{x_j}))^t\Phi(\mathbf{x_i}) + b) - 1 + \xi_i) + C\sum_{i=1}^{M}\xi_i + \sum_{l=1}^{K} d_l\sigma_l - \sum_{i=1}^{M}\beta_i\xi_i - \sum_{l=1}^{K}\gamma_l d_l - \sum_{l=1}^{K} d_l\boldsymbol{\delta}^t\mathbf{A_l} + \mathbf{p^t}\boldsymbol{\delta}$$

Simplyfying further,

$$= \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y_i y_j\Phi(\mathbf{x_i})^t\Phi(\mathbf{x_j}) - \sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y_i y_j\Phi(\mathbf{x_i})^t\Phi(\mathbf{x_j}) - b\sum_{i=1}^{M}\alpha_i y_i + \sum_{i=1}^{M}\alpha_i - \sum_{i=1}^{M}\alpha_i\xi_i + C\sum_{i=1}^{M}\xi_i - \sum_{i=1}^{M}\beta_i\xi_i + \sum_{l=1}^{K} d_l\sigma_l - \sum_{l=1}^{K}\gamma_l d_l - \sum_{l=1}^{K} d_l\boldsymbol{\delta}^t\mathbf{A_l} + \mathbf{p^t}\boldsymbol{\delta}$$

Using equation (B.8)

$$= -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y_i y_j\Phi(\mathbf{x_i})^t\Phi(\mathbf{x_j}) - 0 + \sum_{i=1}^{M}\alpha_i - \sum_{i=1}^{M}(\alpha_i + \beta_i - C)\xi_i + \sum_{l=1}^{K} d_l\sigma_l - \sum_{l=1}^{K}\gamma_l d_l - \sum_{l=1}^{K} d_l\boldsymbol{\delta}^t\mathbf{A_l} + \mathbf{p^t}\boldsymbol{\delta}$$

Substituting equation (B.9) and (B.10)

$$= -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y_i y_j\Phi(\mathbf{x_i})^t\Phi(\mathbf{x_j}) + \sum_{i=1}^{M}\alpha_i - 0 + \sum_{l=1}^{K} d_l(\gamma_l + \boldsymbol{\delta}^t\mathbf{A_l} + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y_i y_j\Phi_l(\mathbf{x_i})^t\Phi_l(\mathbf{x_i})) - \sum_{l=1}^{K}\gamma_l d_l - \sum_{l=1}^{K} d_l\boldsymbol{\delta}^t\mathbf{A_l} + \mathbf{p^t}\boldsymbol{\delta}$$

Finally, the dual objective function is

$$Q = \sum_{i=1}^{M}\alpha_i + \mathbf{p^t}\boldsymbol{\delta} \tag{B.16}$$

$$= \mathbf{1^t}\boldsymbol{\alpha} + \mathbf{p^t}\boldsymbol{\delta} \tag{B.17}$$

For the constraints part,
Similar to SVM, from equations (B.9), (B.13) we have

$$0 \leq \alpha_i \leq C \tag{B.18}$$

$$0 \leq \delta_l \tag{B.19}$$

Equation (B.8) is rewritten as,

$$\mathbf{1}^t Y \boldsymbol{\alpha} = 0 \tag{B.20}$$

Equation (B.10) is rewritten as,

$$\sigma_l = \gamma_l + \boldsymbol{\delta}^t \mathbf{A_l} + \frac{1}{2}\boldsymbol{\alpha^t} \mathbf{Y K_l Y} \boldsymbol{\alpha} \tag{B.21}$$

$$\frac{1}{2}\boldsymbol{\alpha^t} \mathbf{Y K_l Y} \boldsymbol{\alpha} + \gamma_l = \sigma_l - \boldsymbol{\delta}^t \mathbf{A_l} \tag{B.22}$$

$$\frac{1}{2}\boldsymbol{\alpha^t} \mathbf{Y K_l Y} \boldsymbol{\alpha} \leq \sigma_l - \boldsymbol{\delta}^t \mathbf{A_l} \tag{B.23}$$

So, dual formulation is, maximize

$$\begin{gathered} Q_d = \mathbf{1^t}\boldsymbol{\alpha} + \mathbf{p^t}\boldsymbol{\delta} \\ \text{subject to} \quad 0 \leq \boldsymbol{\alpha} \leq C, \quad 0 \leq \boldsymbol{\delta}, \quad \mathbf{1}^t \mathbf{Y} \boldsymbol{\alpha} = 0 \\ \frac{1}{2}\boldsymbol{\alpha^t} \mathbf{Y K_l Y} \boldsymbol{\alpha} \leq \sigma_l - \boldsymbol{\delta^t} \mathbf{A_l} \end{gathered} \tag{B.24}$$

Bibliography

[1] N. Cristianini, A. Elisseeff, J. Shawe-Taylor, and J. Kandla, "On kernel target alignment," in *NIPS*, 2001.

[2] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *JMLR*, vol. 5, pp. 27–72, 2004.

[3] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc of the Neural Information Processing Systems (NIPS)*, 2004.

[4] S. Sonnenburg, G. Raetsch, C. Schaefer, and B. Schoelkopf, "Large scale multiple kernel learning," *JMLR*, vol. 7, pp. 1531–1565, 2006.

[5] A. Rakotomamonjy, F. Bach, Y. Grandvalet, and S. Canu, "Simple mkl," *JMLR*, vol. 9, pp. 2491–2521, 2008.

[6] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[7] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Tech. Rep. 7694, Caltech, 2007.

[8] C. H. Lampert, "Kernel methods in computer vision," *Foundations and Trends in Computer Graphics and Vision*, vol. 4, pp. 193–285, 2009.

[9] P. V. Gehler, *Kernel Learning Approaches for Image Classification*. PhD thesis, Saarland University, Department of Computer Science, 2009.

[10] B. Moghaddam and M. H. Yang, "Learning gender with support faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 707–711, 2002.

[11] G. Andrew and J. Gao, "Scalable training of L_1-regularized log-linear models," in *ICML*, 2007.

[12] A. B. Chan, N. Vasconcelos, and G. Lanckriet, "Direct convex relaxations of sparse SVM," in *ICML*, 2007.

[13] G. Fung and O. L. Mangasarian, "A feature selection newton method for support vector machine classification," Tech. Rep. 02-03, Univ. of Wisconsin, 2002.

[14] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[15] E. E. Boser, I. N. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT*, (New York,USA), pp. 144–152, 1992.

[16] C. Cortes and V. Vapnik, "Support-vector networks," in *Machine Learning*, pp. 273–297, 1995.

[17] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *In Proceedings of the 15th International Conference on Machine Learning*, pp. 515–521, Morgan Kaufmann, 1998.

[18] V. N. Vapnik, "Statistical learning theory," in *Wiley-Interscience*, 1998.

[19] B. Scholkopf, A. S. Klaus, and R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," 1998.

[20] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers, "Fisher discriminant analysis with kernels," in *IEEE Signal Processing Society Workshop In Neural Networks for Signal Processing IX*, pp. 41–48, 1999.

[21] A. Ben-Hur and W. S. Noble, "Kernel methods for predicting protein-protein interactions," in *ISMB (Supplement of Bioinformatics)*, pp. 38–46, 2005.

[22] Furey, Cristianini, Duffy, Bednarski, Schummer, and Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *BIOINF: Bioinformatics*, vol. 16, 2000.

[23] B. Schölkopf, I. Guyon, and J. Weston, "Statistical learning and kernel methods in bioinformatics," 2000.

[24] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[25] P. Baldi, *Bioinformatics: The Machine Learning Approach*. Adaptive Computation and Machine Learning, Cambridge, MA-London: The MIT Press, 2001.

[26] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Kluwer, 1993.

[27] A. M. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of IEEE*, vol. 90, pp. 1151–1163, 2002.

[28] C. H. Lampert, "Kernel methods in computer vision," *Foundations and Trends in Computer Graphics and Vision*, vol. 4, no. 3, pp. 193–285, 2009.

[29] W. M. Campbell, "Generalized linear discriminant sequence kernels for speaker recognition," 2002.

[30] C. Watkins, "Dynamic alignment kernels," Tech. Rep. CSD-TR-98-11, Department of Computer Science, University of London, Egham, England, 1999.

[31] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech & Language*, vol. 20, no. 2-3, pp. 210–229, 2006.

[32] T. Gärtner, "A survey of kernels for structured data," *SIGKDD Explorations*, vol. 5, no. 1, pp. 49–58, 2003.

[33] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*, (East Sussex - San Francisco), pp. 186–195, Morgan Kaufmann, 1998.

[34] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," Tech. Rep. LS VIII-Report, Universität Dortmund, Dortmund, Germany, 1997.

[35] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. KDD*, 2002.

[36] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. CUP, 2004.

[37] S. Qiu and T. Lane, "A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 6, no. 2, pp. 190–199, 2009.

[38] S. Sonnenburg, G. Rch, and C. Schr, "Learning interpretable svms for biological sequence classification," in *BMC Bioinformatics*, pp. 389–407, Springer-Verlag, 2005.

[39] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *ICCV*, 2007.

[40] A. Kembhavi, B. Siddiquie, R. Miezianko, S. McCloskey, and L. Davis, "Scene it or not? incremental multiple kernel learning for object detection," 2009.

[41] S. Nakajima, A. Binder, C. Muller, W. Wojcikiewicz, M. Kloft, U. Brefeld, K.-R. Muller, and M. Kawabe, "Multiple kernel learning for object classification," in *Proceedings of the 12th Workshop on Information-based Induction Sciences*, 2009.

[42] C. Longworth and M. Gales, "Multiple kernel learning for speaker verification," in *ICASSP*, 2008.

[43] Subrahmanya, Niranjan, and Y. C. Shin, "Sparse multiple kernel learning for signal processing applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 788–798, 2010.

[44] D. P. Lewis, T. Jebara, and W. S. Noble, "Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure," *Bioinformatics*, vol. 22, no. 22, pp. 2753–2760, 2006.

[45] F. Yan, J. Kittler, K. Mikolajczyk, and A. Tahir, "Non-sparse multiple kernel learning for fisher discriminant analysis," in *ICDM '09: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, (Washington, DC, USA), pp. 1064–1069, IEEE Computer Society, 2009.

[46] S. jean Kim, A. Magnani, and S. Boyd, "Optimal kernel selection in kernel fisher discriminant analysis," 2006.

[47] I. T. Jolliffe, "Pricipal component analysis," *Springer*, 1986.

[48] J. Makhoul, "Linear prediction : A tutorial review," vol. 63, no. 4, pp. 561–580, 1975.

[49] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms*. 2001.

[50] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines: and other kernel based learning methods*. 2000.

[51] B. Scholkopf and A. J. Smola, *Learning with Kernels:Support Vector Machines, Regularization, Optimization , and Beyond*. 2001.

[52] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems.," in *COLT*, 2000.

[53] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[54] J. Weston and C. Watkins, "Multi-class support vector machines," 22 1998.

[55] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification," in *NIPS* (S. A. Solla, T. K. Leen, and K.-R. Müller, eds.), pp. 547–553, The MIT Press, 1999.

[56] E. Mayoraz and E. Alpaydin, "Support vector machines for multi-class classification," in *IWANN (2)* (J. Mira and J. V. Sánchez-Andrés, eds.), vol. 1607 of *Lecture Notes in Computer Science*, pp. 833–842, Springer, 1999.

[57] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," Tech. Rep. LS VIII-Report, Universität Dortmund, Dortmund, Germany, 1997.

[58] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems*, vol. 11, 1998.

[59] C. Watkins, "Dynamic alignment kernels," Tech. Rep. CSD-TR-98-11, Department of Computer Science, University of London, Egham, England, 1999.

[60] C. S. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: A string kernel for SVM protein classification," in *Pacific Symposium on Biocomputing*, pp. 566–575, 2002.

[61] D. Haussler, "Convolution kernels on discrete structure," Tech. Rep. UCSC-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

[62] R. I. Kondor and J. D. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," in *ICML* (C. Sammut and A. G. Hoffmann, eds.), pp. 315–322, Morgan Kaufmann, 2002.

[63] J.-P. Vert, "A tree kernel to analyse phylogenetic profiles," in *ISMB (Supplement of Bioinformatics)*, pp. 276–284, 2002.

[64] E. Takimoto and M. K. Warmuth, "Path kernels and multiplicative updates," *Journal of Machine Learning Research*, vol. 4, pp. 773–818, 2003.

[65] T. Jebara, R. I. Kondor, and A. Howard, "Probability product kernels," *Journal of Machine Learning Research*, vol. 5, pp. 819–844, 2004.

[66] A. Barla, F. Odone, and A. Verri, "Hausdorff kernel for 3D object acquisition and detection," in *ECCV*, p. IV: 20 ff., 2002.

[67] S. Boughorbel, J. P. Tarel, and N. Boujemaa, "Generalized histogram intersection kernel for image recognition," in *ICIP*, pp. III: 161–164, 2005.

[68] M. N. S. S. K. P. Kumar and C. V. Jawahar, "Configurable hybrid architectures for character recognition applications," in *ICDAR*, vol. 1, pp. 1199–1203., 2005.

[69] H. Jong and S. P. Milanfar, "Generic human action recognition from a single example," 2009.

[70] S. Dambreville, Y. Rathi, and A. Tannenbaum, "Shape-based approach to robust image segmentation using kernel PCA," in *CVPR*, pp. I: 977–984, 2006.

[71] D. R. Heisterkamp, J. Peng, and H. K. Dai, "Adaptive quasiconformal kernel metric for image retrieval," in *CVPR*, pp. II:388–393, 2001.

[72] D. G. Lowe, "Object recognition from local scale-invariante features," in *Proc 7th Int Conf on Computer Vision, Corfu, Greece*, 1999.

[73] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, April 2002.

[74] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondence," in *Proc IEEE Conf on Computer Vision and Pattern Recognition, San Diego CA, June 20-25*, 2005.

[75] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1265–1278, August 2005.

[76] M. Varma and A. Zisserman, "Classifying images of materials: Achieving viewpoint and illumination independence," in *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, vol. 3, pp. 255–271, Springer-Verlag, May 2002.

[77] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *IEEE Conference on Computer Vision and Pattern Recognition 2007 (CVPR'07)*, June 2007.

[78] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *Int Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001.

[79] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *In ICCV*, pp. 1458–1465, 2005.

[80] S. Lazebnik, C. Schmid, and J. Ponce, "A maximum entropy framework for part-based texture and object recognition," in *ICCV*, pp. I: 832–838, 2005.

[81] S. Belongie, C. C. Fowlkes, F. Chung, and J. Malik, "Spectral partitioning with indefinite kernels using the nyström extension," in *ECCV*, p. III: 531 ff., 2002.

[82] S.Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *In CVPR*, pp. 2169–2178, 2006.

[83] F. R. Bach, "Exploring large feature spaces with hierarchical multiple kernel learning," in *Proc of the Neural Information Processing Systems (NIPS)*, 2008.

[84] P. Gehler and S. Nowozin, "Infinite kernel learning," in *In NIPS 2008 Workshop on "Kernel Learning: Automatic Selection of Optimal Kernels"*, (Whistler, Canada), 2008.

[85] C. Burges and B. Scholkopf, "Improving the accuracy and speed of support vector machines," in *NIPS*, vol. 9, p. 375, 1997.

[86] G. Wu, E.Chang, Y. Chen, and C.Hughes, "Incremental approximate matrix factorization for speeding up suppert vectro machines," in *SIGKDD*, (New York, NY, USA), pp. 760–766, 2006.

[87] K. Crammer, J. Keshet, and Y. Singer, "Kernel design using boosting," in *Proc of the Neural Information Processing Systems (NIPS)*, pp. 537–544, 2002.

[88] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola, "On kernel-target alignment," in *Proc of the Neural Information Processing Systems (NIPS)*, 2001.

[89] A. Argyriou, C. A. Micchelli, and M. Pontil, "Learning convex combinations of continuously parameterized basic kernels," in *COLT*, 2005.

[90] C. S. Ong, A. J. Smola, and R. C. Williamson, "Learning the kernel with hyperkernels," *JMLR*, vol. 6, pp. 1043–1071, 2005.

[91] I. W. Tsang and J. T. Kwok, "Efficient hyperkernel learning using second-order cone programming," *IEEE Trans. Neural Networks*, vol. 17, no. 1, pp. 48–58, 2006.

[92] A. Zien and C. S. Ong, "Multiclass multiple kernel learning," in *ICML*, pp. 1191–1198, 2007.

[93] M. Avriel, *Nonlinear Programming: Analysis and methods*. Englewood Cliffs: Prentice-Hall Inc., 1976.

[94] M. Gönen and E. Alpaydin, "Localized multiple kernel learning," in *ICML*, (New York, NY, USA), pp. 352–359, ACM, 2008.

[95] S. N. Jagarlapudi, D. G, R. S, C. Bhattacharyya, A. Ben-Tal, and R. K.R., "On the algorithmics and applications of a mixed-norm based kernel learning formulation," in *Advances in Neural Information Processing Systems 22* (Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, eds.), pp. 844–852, 2009.

[96] M. Kowalski, M. Szafranski, and L. Ralaivola, "Multiple indefinite kernel learning with mixed norm regularization," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, (New York, NY, USA), pp. 545–552, ACM, 2009.

[97] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien, "Efficient and accurate lp-norm multiple kernel learning," in *Advances in Neural Information Processing Systems 22* (Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, eds.), pp. 997–1005, 2009.

[98] S. Ji, L. Sun, R. Jin, and J. Ye, "Multi-label multiple kernel learning," in *Advances in Neural Information Processing Systems 21* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), pp. 777–784, 2009.

[99] M. Kloft, U. Brefeld, P. Laskov, and S. Sonnenburg, "Non-sparse Multiple Kernel Learning," in *NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

[100] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for Support Vector Machines," *Machine Learning*, vol. 46, pp. 131–159, 2002.

[101] J. M. Danskin, *The Theorey of Max-Min and its Applications to Weapons Allocation Problems*. 1967.

[102] C. Lampert and M. Blaschko, "A multiple kernel learning approach to joing multi-class object detection.," in *DAGM*, 2008.

[103] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[104] L. Song, A. Smola, A. Gretton, K. Borgwardt, and J. Bedo, "Supervised feature selection via dependence estimation," in *ICML*, pp. 823–830, 2007.

[105] J. Bi, T. Zhang, and K. P. Bennett, "Column-generation boosting methods for mixture of kernels," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 521–526, ACM, 2004.

[106] S. Baluja and H. Rowley, "Boosting sex identification performance," *Int Journal of Computer Vision*, vol. 71, no. 1, pp. 111–119, 2007.

[107] M. Lustig, "Sparse mri," *PhD thesis Stanford*, 2008.

[108] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *Int Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43–72, 2005.

[109] A. Ferencz, E. Learned-Miller, and J. Malik, "Learning to locate informative features for visual identification," *Int Journal of Computer Vision*, vol. 77, no. 1, pp. 3–24, 2008.

[110] A. Frome, Y. Singer, and J. Malik, "Image retrieval and recognition using local distance functions," in *Proc of the Neural Information Processing Systems (NIPS)*, pp. 417–424, 2006.

[111] A. Frome, Y. Singer, F. Sha, and J. Malik, "Learning globally-consistent local distance functions for shape-based image retrieval and classification," in *ICCV*, pp. 1–8, 2007.

[112] D. Gao and N. Vasconcelos, "Bottom-up saliency is a discriminant process," in *ICCV*, 2007.

[113] F. Moosmann, D. Larlus, and F. Jurie, "Learning saliency maps for object categorization," in *Int. Workshop on The Representation and Use of Prior Knowledge in Vision*, 2006.

[114] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc of the IEEE Conf on Computer Vision and Pattern Recognition*, vol. 2, (New York, New York), pp. 2169–2178, 2006.

[115] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[116] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large databases for recognition," in *Proc of the IEEE Conf on Computer Vision and Pattern Recognition*, 2008.

[117] A. Berg and J. Malik, "Geometric blur for template matching," in *Proc of the IEEE Conf on Computer Vision and Pattern Recognition*, vol. 1, pp. 607–614, 2001.

[118] F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition," in *ICCV*, 2005.

[119] O. Chum and A. Zisserman, "An exemplar model for learning object classes," in *Proc of the IEEE Conf on Computer Vision and Pattern Recognition*, 2007.

[120] B. Fulkerson, A. Vedaldi, and S. Soatto, "Localizing objects with smart dictionaries.," in *Proc European Conf on Computer Vision*, 2008.

[121] J. Winn, A. Criminisi, and T. Minka, "Gradient-based learning applied to document recognition," in *ICCV*, 2005.

[122] O. due Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition - a survey," in *Patter Recognition*, pp. 641–655, 1996.

[123] S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. Joshi, "Text extraction and document image segmentation using matched wavelets and mrf model," *IEEE Transactions on Image Processing*, vol. 16, pp. 2117–2128, August 2007.

[124] A. Krempp, D. Geman, and Y. Amit, "Sequential learning of reusable parts for object detection," tech. rep., Computer Science Department, Johns Hopkins University, 2002.

[125] P. Clark and M. Mirmehdi, "Recognising text in real scenes," *International Journal on Document Analysis and Recognition*, vol. 4, pp. 243–257, 2002.

[126] M. S. Brown, M. Sun, R. Yang, L. yun, and W. B. Seales, "Restoring 2d content from distorted documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[127] K. Kise and D. S. Doermann, eds., *Proceedings of the Second International Workshop on Camera-based Document Analysis and Recognition CBDAR*, (Curitiba, Brazil), September 22 2007. http://www.imlab.jp/cbdar2007/.

[128] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: A comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 63–84, January 2000.

[129] U. Pal, N. Sharma, T. Wakabayashi, and F. Kimura, "Off-line handwritten character recognition of devnagari script," in *International Conference on Document Analysis and Recognition (ICDAR)*, (Curitiba, PR, Brazil), pp. 496–500, IEEE, September 23-26 2007.

[130] Z. Tu, X. Chen, A. L. Yuille, and S. C. Zhu, "Image parsing: Unifying segmentation, detection, and recognition," *International Journal of Computer Vision, Marr Prize Issue*, 2005.

[131] Y. Jin and S. Geman, "Context and hierarchy in a probabilistic image model," in *Proc IEEE Conf on Computer Vision and Pattern Recognition, New York NY, June 17-22*, 2006.

[132] J. J. Weinman and E. Learned Miller, "Improving recognition of novel input with similarity," in *Proc IEEE Conf on Computer Vision and Pattern Recognition, New York NY, June 17-22*, 2006.

[133] Y. le Cun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November 1998.

[134] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *Proc IEEE Conf on Computer Vision and Pattern Recognition, New York NY, June 17-22*, 2006.

[135] F.Jurie and B.Triggs, "Creating efficient code books for visual recognition.," in *Proceedings of the IEEE International Conference on Computer Vision*, 2005.

[136] J.J. Weinman and E.Learned Miller, "Improving recognition of novel input with similarity," in *In Proc IEEE Conf on Computer Vision and Pattern Recognition,*, (New York), 2006.

[137] A. E. Johnson and M. Herbert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.