LOCALIZED MULTIPLE KERNEL ALGORITHMS FOR MACHINE LEARNING

by

Mehmet Gönen

B.S., Industrial Engineering, Boğaziçi University, 2003

M.S., Computer Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2010

To my lovely father Tahir Gönen ...

ACKNOWLEDGEMENTS

I would first like to thank my supervisor Prof. Ethem Alpaydın for all the support he has given me throughout my graduate education; I have been extremely fortunate. Much of his energy, motivation and of the long discussions we had can be found in this thesis. I am very grateful to my examiners, Prof. H. Levent Akın, Prof. İ. Kuban Altınel, Prof. Türkan Haliloğlu, and Assoc. Prof. Berrin Yanıkoğlu, for taking the time to give me such useful feedback.

I am also very grateful to have been a Ph.D. student at the Department of Computer Engineering, which always provided me with a lively and pleasant atmosphere in which to work. I would like to thank all the people I have met and worked with during my time at the department. It is not possible to thank everyone here but I would particularly like to mention Prof. Cem Ersoy, Assist. Prof. A. Taylan Cemgil, and my longtime friends, Salim Eryiğit, Onur Dikmen, Aydın Ulaş, İsmail Arı, Itır Karaç, and Oya Aran, for their scientific inputs, discussions, and friendship over the years. Most of the work in this thesis has been performed at the Perceptual Intelligence Laboratory (PILAB). I deeply appreciate the support of all current and former members of PILAB. Thanks to you all for bearing me all the time!

I would like to thank my parents and primary school teachers Ümmü and Tahir Gönen, and all my family who have supported me all these years, even from hundreds of kilometers away. Finally, I would like to thank my dearest love Gülefşan for making me the happiest men in the world after she walked into my life.

This thesis has been supported by Boğaziçi University Scientific Research Project 07HA101, the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant EEEAG 107E222, and the Ph.D. scholarship (2211) from TÜBİTAK.

ABSTRACT

LOCALIZED MULTIPLE KERNEL ALGORITHMS FOR MACHINE LEARNING

In recent years, several multiple kernel learning methods have been proposed in the machine learning literature. Different kernels correspond to different notions of similarity and multiple kernel learning can be used to combine them. It can also be used to integrate different inputs coming from different representations, possibly from different sources or modalities, by combining kernels calculated on these representations. This thesis contains a number of extensions to the original multiple kernel learning framework, together with experimental results that support their utility on benchmark data sets from the UCI Machine Learning Repository as well as several image image recognition and bioinformatics data sets.

This thesis introduces a regularized multiple kernel learning framework and proposes to use the response surface methodology to search for the best regularization parameter set using validation data. Optimizing such regularization parameters allows us to obtain more robust decision functions for the classification task at hand. Kernels that do not help increase the classification accuracy are pruned by selecting their regularization parameters accordingly, obtaining smoother discriminants. Eliminating some of the kernels directly or decreasing the number of stored support vectors reduces the testing time for new instances.

This thesis also proposes a cost-conscious strategy to include the cost of kernel computations and data acquisition/generation into the multiple kernel learning framework. The results show that incorporating a cost factor into the model enables us to use only the necessary kernels, avoiding costly kernel computations and input generation for some data representations in the testing phase, when possible.

The main contribution of this thesis is formulation of a localized multiple kernel learning framework that is composed of a kernel-based learning algorithm and a gating model to assign data-dependent weights to kernel functions. We derive the learning algorithm for three different gating models and apply localized multiple kernel learning to binary classification, regression, multiclass classification, and one-class classification problems. For classification problems that use different feature representations, our proposed method is able to construct better classifiers by combining the kernels on these representations locally. This localized formulation achieves higher average test accuracies and stores fewer support vectors compared to the canonical multiple kernel combination with global weights. We also see that, as expected, combining heterogeneous feature representations is more advantageous than combining multiple copies of the same representation. For image recognition problems, our proposed method identifies the relevant parts of each input image separately by using the gating model as a saliency detector on the kernels calculated on the image patches. Different from the multiple kernel learning methods proper, our proposed method can combine multiple copies of the same kernel. We show that even if we provide more kernels than needed, our proposed approach uses only as many support vectors as required and does not overfit.

We also introduce a supervised and localized dimensionality reduction method that trains local projection kernels coupled with a kernel-based learning algorithm. On visualization tasks, our proposed method is able to maintain the multimodality of a class by placing clusters of the same class on the same side of the hyperplane while preserving a separation between them. On classification tasks, it achieves better results than other methods by attaining both higher test accuracy and storing fewer support vectors due to the coupled optimization of the discriminant and the local projection matrices used in dimensionality reduction.

ÖZET

YAPAY ÖĞRENME İÇİN YEREL ÇOKLU ÇEKİRDEK ALGORİTMALARI

Son yıllarda yapay öğrenme için çeşitli çoklu çekirdek yöntemleri önerilmiştir. Değişik çekirdekler değişik benzerlik ölçütleri tanımlamaktadır, ve çoklu çekirdek öğrenimi farklı benzerlik ölçütlerini birleştirmek için kullanılabildiği gibi farklı veri gösterimleri üzerinde hesaplanan çekirdek fonksiyonlarını kullanarak farklı kaynaklardan gelen veya farklı özelliklerdeki verileri birleştirmek için de kullanılabilir. Bu tez, çoklu çekirdek öğrenimi için yeni yöntemler önermekte ve bu yöntemlerin kullanılabilirliğini standart karşılaştırma veri kümelerinin yanısıra görüntü tanıma ve biyoinformatik veri kümeleri üzerinde alınan deneysel sonuçlarla desteklemektedir.

Bu tez, çoklu çekirdek öğreniminde düzenli sonuçlar elde etmek için, tepki yüzeyi yöntemini kullanarak, geçerleme verisi üzerinde en iyi düzen parametrelerinin seçimi için yeni bir çoklu çekirdek öğrenim yöntemi önermektedir. Düzen parametrelerinin eniyilenmesi elimizdeki sınıflandırma problemi için daha iyi karar fonksiyonları elde etmemizi sağlamaktadır. Sınıflandırma başarısına katkıda bulunmayan çekirdekler düzen parametrelerinin bu doğrultuda seçilmesiyle elenmekte ve daha iyi ayırtaçlar elde edilmektedir. Bazı çekirdelerin kullanılmaması veya daha az destek vektörü saklanması ile yeni örnekler için deneme zamanı azalmaktadır.

Bu tez, aynı zamanda, çekirdek hesaplama ve veri toplama/işleme maliyetlerini dikkate alan maliyet-bilinçli bir çoklu çekirdek öğrenimi yöntemi önermektedir. Sonuçlar maliyet etkeninin modele eklenmesinin deneme aşamasında yalnız gerekli çekirdeklerin kullanılabilmesini ve bazı veri gösterimleri için maliyetli çekirdek hesaplamalarından ve veri üretim aşamasından kaçınılabilmesini sağladığını göstermektedir.

Bu tezin ana katkısı, çekirdek-tabanlı bir öğrenme algoritması ve çekirdek fonksiyonlarına veriye bağlı ağırlıklar atayan bir geçit modelinden oluşan yerel çoklu çekirdek öğrenim yöntemi önermesidir. Öğrenme algoritmasını üç değişik geçit modeli için geliştirdik ve yerel çoklu çekirdek öğrenimini iki sınıflı sınıflandırma, regresyon, çok sınıflı sınıflandırma ve tek sınıflı sınıflandırma problemlerine uyguladık. Önerilen yöntem, değişik veri gösterimleri üzerinde tanımlanan sınıflandırma problemlerinde, bu veri gösterimleri üzerinde hesaplanan çekirdekleri yerel olarak birleştirerek daha iyi sınıflandırıcılar üretmektedir. Bilinen çoklu çekirdek öğrenim yöntemiyle karşılaştırıldığında, önerdiğimiz yöntem daha yüksek ortalama sınıflandırma başarısı elde etmekte ve daha az destek vektörü saklamaktadır. Beklendiği gibi, farklı veri gösterimlerinin birleştirilmesinin aynı veri gösteriminin birçok kopyasının birleştirilmesine göre daha üstün olduğunu gördük. Önerilen yöntem, görüntü tanıma problemlerinde geçit modeli ile görüntü parçaları üzerinde hesaplanan çekirdekler içinden seçim yaparak her bir örnek görüntünün belirgin parçalarını bulabilmektedir. Ayrıca, genel çekirdek ağırlıkları kullanan yöntemlerden farklı olarak aynı çekirdeğin birçok kopyasını birleştirebilmektedir. Gerekenden fazla çekirdek verildiği durumda bile, modelin gerektiği kadar destek vektörü kullandığını ve aşırı öğrenmediğini gösterdik.

Yerel izdüşüm çekirdekleri öğrenen ve çekirdek-tabanlı öğrenme algoritmaları ile birleşik, gözetimli ve yerel bir boyut azaltma yöntemi önerdik. Bu yöntem, görselleştirme görevlerinde bir sınıfın çoklu biçimli yapısını, bu sınıfın örneklerini ayırtacın aynı tarafına koyarak ve aralarındaki ayrımı koruyarak sürdürebilmektedir. Sınıflandırma görevlerinde ayırtaç parametlerinin ve boyut azaltmada kullanılan yerel izdüşüm matrislerinin birlikte eniyilenmesiyle, diğer yöntemlere göre daha yüksek sınıflandırma başarısı elde edilmekte ve daha az destek vektörü saklanmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\odot$	Hadamard product
$\|\cdot\|_p$	l_p-norm
$\|\cdot\|_F$	Frobenius norm
$\top$	Transpose
$\langle\cdot,\cdot\rangle$	Dot product
$1(\cdot)$	1 if the parameter is true, 0 otherwise
b	Bias term
C	Trade-off parameter
d	Regularization parameter
D	Dimensionality of the original feature space
$k(\cdot,\cdot)$	Kernel function
K	Number of classes
$\mathbf{K}$	Kernel matrix
N	Number of training instances
$\mathbb{N}$	Natural numbers
$\mathcal{N}(\cdot,\cdot)$	Normal distribution
P	Number of kernel to be combined
$\mathbb{R}$	Real numbers
$\mathbb{R}_+$	Nonnegative real numbers
$\mathbb{R}_{++}$	Positive numbers
S	Dimensionality of the mapped feature space
$\operatorname{tr}(\cdot)$	Trace
$\boldsymbol{x}$	Data instance
$\mathbf{V}$	Gating model parameters
$\boldsymbol{w}$	Weight coefficients
$\mathbf{W}$	Projection matrix
y	Output value
$\mathbb{Z}_+$	Nonnegative integers

α	Support vector coefficient
δ_i^j	Kronecker delta: 1 if $i = j$, 0 otherwise
ϵ	Error tube width
η	Kernel weight
ξ	Slack variable
$\Phi(\cdot)$	Mapping function

CYGD	Comprehensive Yeast Genome Database
DLLE	Discriminant Locally Linear Embedding
FDA	Fisher Discriminant Analysis
GMKL	Generalized Multiple Kernel Learning
GPK	Global Projection Kernels
KPCA	Kernel Principal Component Analysis
LFDA	Local Fisher Discriminant Analysis
LLE	Locally Linear Embedding
LLP	Local Learning Projections
LMKL	Localized Multiple Kernel Learning
LP	Linear Programming
LPK	Local Projection Kernels
LPP	Locality Preserving Projections
MCPPSVM	Multiclass Posterior Probability Support Vector Machine
MCSVM	Multiclass Support Vector Machine
MKL	Multiple Kernel Learning
MSE	Mean Square Error
OCSVM	One-Class Support Vector Machine
PAM	Point Accepted Mutation
PCA	Principal Component Analysis
PDB	Protein Data Bank
PPSVM	Posterior Probability Support Vector Machine
QCQP	Quadratically Constrained Quadratic Programming
QP	Quadratic Programming

RMKL	Regularized Multiple Kernel Learning
RSM	Response Surface Methodology
SDP	Semidefinite Programming
SILP	Semi-Infinite Linear Programming
SMKL	Sparse Multiple Kernel Learning
SOCP	Second-Order Cone Programming
SVM	Support Vector Machine
SVR	Support Vector Regression

1. INTRODUCTION

Machine learning tries to find structural properties in the observed data and aims to generate rules to make predictions for unseen data. In supervised learning, we are given observed data instances with their outputs, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, and the aim is to estimate the output, y, for an unseen test instance, $\boldsymbol{x}$. $\{y_i\}_{i=1}^N$ correspond to class codes in classification and to output values in regression. Table 1.1 lists some example machine learning problems with their input representations, features, and outputs. For example, in face recognition problems, we are given an input face image and we want to identify the person in the image using the pixel values as the features.

Table 1.1. Example machine learning problems.

Problem	Input	Features	Output
Digit recognition		Pixel values	$\{0, 1, \ldots, 9\}$
Face recognition		Pixel values	{Person1, Person2, ...}
Disease diagnosis	`ATCGGT...TTA`	Nucleotide bases	{Healthy, Not healthy}

In classification, a discriminant is a function that separates the instances of one class from the instances of other classes in the input space and when it is linear, the discriminant function is written as $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$, where $\boldsymbol{w}$ and b are the parameters that we need to learn.

1.1. Binary Classification Support Vector Machines

Support Vector Machine (SVM) is a discriminative binary classifier based on the theory of structural risk minimization (Vapnik, 1998). Given a sample of independent and identically distributed training instances, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^D$

and $y_i \in \{-1, +1\}$ is its class label, SVM finds the linear discriminant with the maximum margin in the feature space induced by the mapping function, $\Phi\colon \mathbb{R}^D \to \mathbb{R}^S$. The discriminant function is

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b$$

whose parameters can be determined by solving the following quadratic optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ \text{with respect to} \quad & \boldsymbol{w} \in \mathbb{R}^S, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to} \quad & y_i(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i) \rangle + b) \geq 1 - \xi_i \qquad \forall i \end{aligned} \tag{1.1}$$

where $\boldsymbol{w}$ is the vector of weight coefficients, S is the dimensionality of the feature space obtained by $\Phi(\cdot)$, C is a predefined positive trade-off parameter between model simplicity and classification error, $\boldsymbol{\xi}$ is the vector of slack variables, and b is the bias term of the separating hyperplane.

We obtain the Lagrangian dual of the primal problem (1.1) as follows:

$$L_D = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i) \rangle + b) - 1 + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i$$

and taking the derivatives of L_D with respect to the primal variables gives

$$\begin{aligned} \frac{\partial L_D}{\partial \boldsymbol{w}} &= 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{N}\alpha_i y_i \Phi(\boldsymbol{x}_i) \\ \frac{\partial L_D}{\partial b} &= 0 \Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \frac{\partial L_D}{\partial \xi_i} &= 0 \Rightarrow C = \alpha_i + \beta_i \qquad \forall i. \end{aligned} \tag{1.2}$$

From L_D and (1.2), the dual formulation is obtained as

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_i y_i y_j \underbrace{\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle}_{k(\boldsymbol{x}_i, \boldsymbol{x}_j)} \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned} \tag{1.3}$$

where $\boldsymbol{\alpha}$ is the vector of dual variables corresponding to each separation constraint and the obtained kernel matrix of $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semidefinite. Solving this, we get $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\boldsymbol{x}_i)$ and the discriminant function can be written as

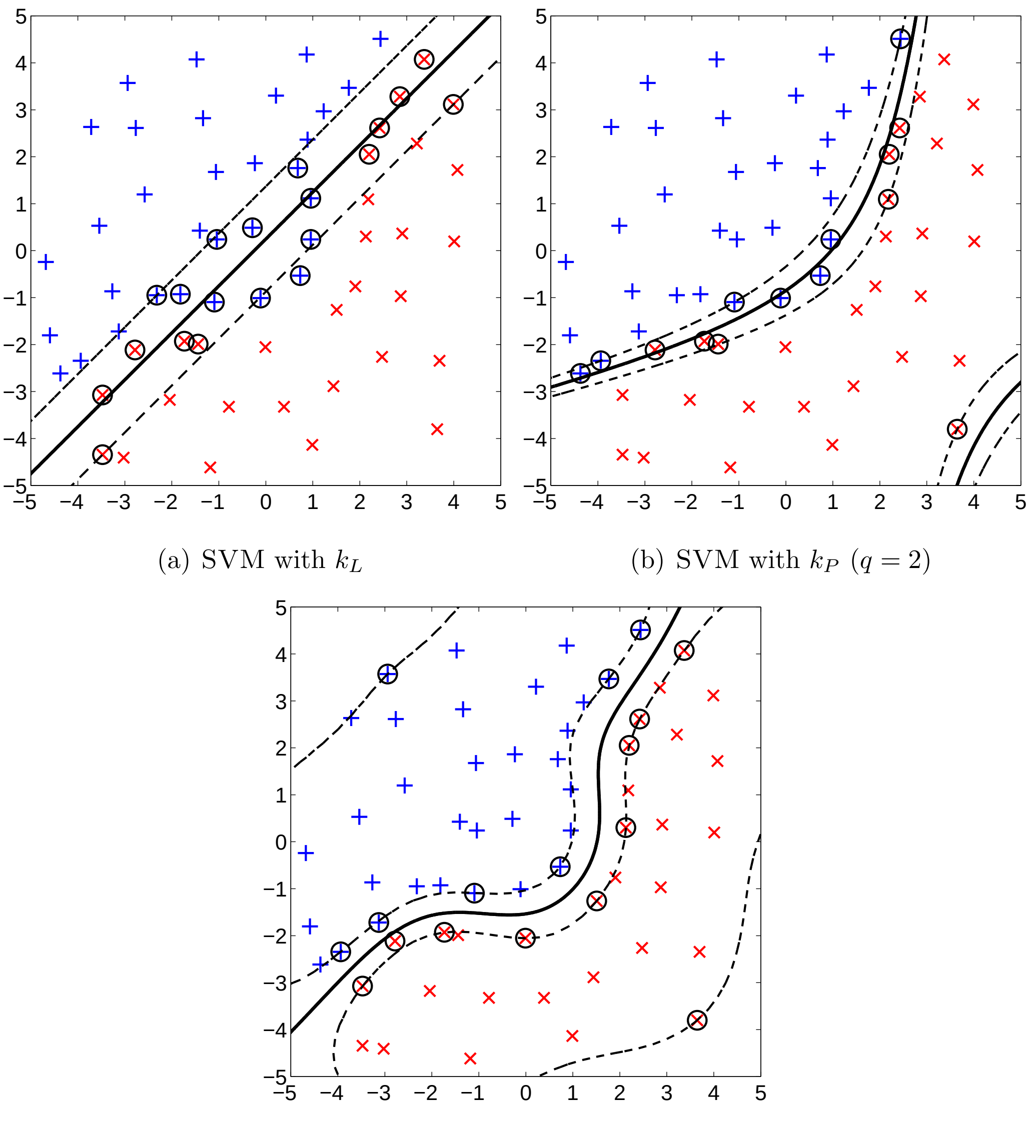
$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i \underbrace{\langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}) \rangle}_{k(\boldsymbol{x}_i, \boldsymbol{x})} + b.$$

The training instances with nonzero α_i values are called *support vectors*. Other training instances do not contribute to the decision function and we do not need to store these instances for the testing phase.

There are several kernel functions successfully used in the literature such as the linear kernel (k_L), the polynomial kernel (k_P), and the Gaussian kernel (k_G):

$$\begin{aligned}
k_L(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\
k_P(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^q \qquad q \in \mathbb{N} \\
k_G(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 / s^2\right) \qquad s \in \mathbb{R}_{++}.
\end{aligned}$$

Figure 1.1 shows the discriminants and support vectors found by SVM with these three kernels on a toy data set. There are also kernel functions proposed for particular applications, such as natural language processing (Lodhi *et al.*, 2002) and bioinformatics (Schölkopf *et al.*, 2004a).



(a) SVM with k_L

(b) SVM with k_P $(q = 2)$

(c) SVM with k_G $(s = 1)$

Figure 1.1. SVM solutions on a toy data set using three different kernels. The solid lines show the discriminants learned and the dashed lines show the margin boundaries. The circled data points represent the support vectors stored.

1.2. Regression Support Vector Machines

In regression problems, we are given real-valued outputs, $y_i \in \mathbb{R}$, instead of $\{-1, +1\}$ class labels. Support Vector Regression (SVR) starts by assuming a linear function in the feature space induced by the mapping function and uses ϵ-insensitive error function in (1.4), that is, it ignores the error value if its magnitude is less than ϵ (Vapnik, 1998):

$$e(y, f(\boldsymbol{x})) = \begin{cases} 0, & \text{if } |y - f(\boldsymbol{x})| \leq \epsilon, \\ |y - f(\boldsymbol{x})|, & \text{otherwise.} \end{cases} \tag{1.4}$$

SVR needs two positive slack variables, ξ_i^+ and ξ_i^-, for each training instance because there is no "correct" side for regression and deviations in both directions are penalized. The training instances that are outside of the ϵ-tube introduce error and the primal optimization problem can be written as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}(\xi_i^+ + \xi_i^-) \\ \text{with respect to} \quad & \boldsymbol{w} \in \mathbb{R}^S, \boldsymbol{\xi}^+ \in \mathbb{R}_+^N, \boldsymbol{\xi}^- \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to} \quad & \epsilon + \xi_i^+ \geq y_i - \langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i)\rangle - b \qquad \forall i \\ & \epsilon + \xi_i^- \geq \langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i)\rangle + b - y_i \qquad \forall i. \end{aligned} \tag{1.5}$$

We obtain the Lagrangian dual of the primal problem (1.5) as follows:

$$\begin{aligned} L_D = \frac{1}{2}\|\boldsymbol{w}\|_2^2 &+ C\sum_{i=1}^{N}(\xi_i^+ + \xi_i^-) - \sum_{i=1}^{N}\alpha_i^+(\epsilon + \xi_i^+ - y_i + \langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i)\rangle + b) \\ &- \sum_{i=1}^{N}\alpha_i^-(\epsilon + \xi_i^- - \langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i)\rangle - b + y_i) - \sum_{i=1}^{N}\beta_i^+\xi_i^+ - \sum_{i=1}^{N}\beta_i^-\xi_i^- \end{aligned}$$

and taking the derivatives of L_D with respect to the primal variables gives

$$\begin{aligned} \frac{\partial L_D}{\partial \boldsymbol{w}} &= 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)\Phi(\boldsymbol{x}_i) \\ \frac{\partial L_D}{\partial b} &= 0 \Rightarrow \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-) = 0 \\ \frac{\partial L_D}{\partial \xi_i^+} &= 0 \Rightarrow C = \alpha_i^+ + \beta_i^+ \qquad \forall i \\ \frac{\partial L_D}{\partial \xi_i^-} &= 0 \Rightarrow C = \alpha_i^- + \beta_i^- \qquad \forall i. \end{aligned} \tag{1.6}$$

From L_D and (1.6), the dual formulation is obtained as

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N} y_i(\alpha_i^+ - \alpha_i^-) - \epsilon \sum_{i=1}^{N}(\alpha_i^+ + \alpha_i^-) \\
& - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
\text{with respect to} \quad & \boldsymbol{\alpha}^+ \in \mathbb{R}_+^N, \boldsymbol{\alpha}^- \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-) = 0 \\
& C \geq \alpha_i^+ \geq 0 \qquad \forall i \\
& C \geq \alpha_i^- \geq 0 \qquad \forall i
\end{aligned}$$

and the estimation function can be written as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

Figure 1.2 shows the fit and support vectors learned by SVR with the Gaussian kernel on a toy data set.

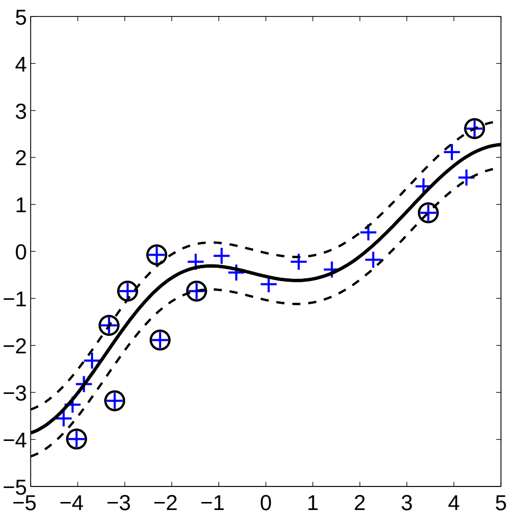


Figure 1.2. SVR solution on a toy data set using k_G ($s = 1$). The solid line shows the fit learned and the dashed lines show the error tube.

1.3. One-Class Support Vector Machines

Kernel machines can also be used for novelty and outlier detection problems. One-Class SVM (OCSVM) is a discriminative method proposed for this purpose and the task is to learn the smoothest hyperplane that puts most of the training instances to one side of the hyperplane while allowing other instances remaining on the other side with a cost (Schölkopf and Smola, 2002). The corresponding primal optimization problem is

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i + b \\ \text{with respect to} \quad & \boldsymbol{w} \in \mathbb{R}^S, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to} \quad & \langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i)\rangle + b + \xi_i \geq 0 \qquad \forall i. \end{aligned} \tag{1.7}$$

We obtain the Lagrangian dual of the primal problem (1.7) as follows:

$$L_D = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i + b - \sum_{i=1}^{N}\alpha_i(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i)\rangle + b + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i$$

and taking the derivatives of L_D with respect to the primal variables gives

$$\begin{aligned} \frac{\partial L_D}{\partial \boldsymbol{w}} &= 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{N}\alpha_i\Phi(\boldsymbol{x}_i) \\ \frac{\partial L_D}{\partial b} &= 0 \Rightarrow \sum_{i=1}^{N}\alpha_i = 1 \\ \frac{\partial L_D}{\partial \xi_i} &= 0 \Rightarrow C = \alpha_i + \beta_i \qquad \forall i. \end{aligned} \tag{1.8}$$

From L_D and (1.8), the dual formulation is obtained as

$$\begin{aligned}
\text{maximize} \quad & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_i k(\boldsymbol{x}_i,\boldsymbol{x}_j) \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N}\alpha_i = 1 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned}$$

and the discriminant function can be written as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N}\alpha_i k(\boldsymbol{x}_i,\boldsymbol{x}) + b.$$

Figure 1.3 shows the discriminant and support vectors found by OCSVM with the Gaussian kernel on a toy data set.

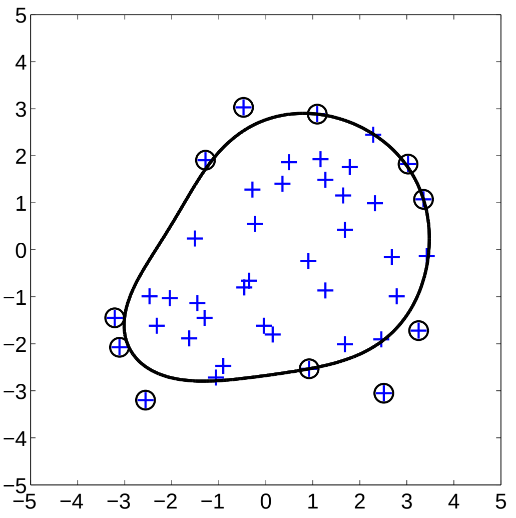


Figure 1.3. OCSVM solution on a toy data set using k_G ($s = 2$). The solid line shows the discriminant learned.

1.4. Multiclass Classification Support Vector Machines

In a multiclass classification problem, we are given a training set where an instance $\boldsymbol{x}_i$ can belong to one of $K > 2$ classes and the class label is given as $y_i \in \{1, \ldots, K\}$. There are two basic approaches in the literature to solve multiclass problems: In the *multimachine approach*, the original multiclass problem is converted to a number of independent, uncoupled two-class problems. In the *single-machine approach*, the constraints due to having multiple classes are coupled in a single formulation.

1.4.1. Multimachine Approaches

Multimachine approaches construct separate, uncoupled two-class problems from the training set. After solving these two-class problems, a voting scheme is required to decide on the estimated class of a given test point. Decomposition step can be performed in three different ways: *one-versus-all* and *all-versus-all.*

<u>1.4.1.1. One-Versus-All Approach.</u> In one-versus-all approach, K distinct binary classifiers (K distinct discriminants) are trained to separate one class from all others (Hsu and Lin, 2002; Rifkin and Klautau, 2004). During test, the class label that is obtained from the binary classifier with the maximum output value is assigned to a test instance. Each binary classifier uses all training samples, and for each class, we have a N-variable Quadratic Programming (QP) problem to be solved. There are two basic concerns about this approach: First, the binary classifiers are trained on many more negative examples than positive ones. Second, the real valued outputs of binary classifiers may be in different scales and direct comparison between them is not applicable (Mayoraz and Alpaydın, 1999).

<u>1.4.1.2. All-Versus-All Approach.</u> Another approach is the all-versus-all or pairwise decomposition (Schmidt and Gish, 1996; Krefsel, 1998), where there are $K(K-1)/2$ binary classifiers for each possible pair of classes. The classifier count is generally much larger than one-versus-all, but when separating two classes, instances of all classes except these two are ignored and hence the quadratic programs in each classifier are much smaller, making it possible to train the system very fast. This approach has the disadvantage of potential variance increase due to the small training set size for each classifier (Lee *et al.*, 2001). The test procedure should utilize a voting scheme to decide which class a test point belongs to, and a modified testing procedure to speed up by

using directed acyclic graph traversals instead of evaluating all $K(K-1)/2$ classifiers is proposed (Platt *et al.*, 2000).

1.4.2. Single-Machine Approaches

A more natural way than using the multimachine approach is to construct a decision function by considering all classes at once (Vapnik, 1998; Weston and Watkins, 1998; Bredensteiner and Bennett, 1999). For the single-machine approach called Multiclass SVM (MCSVM), for class l, we write the discriminant function as follows:

$$f^l(\boldsymbol{x}) = \langle \boldsymbol{w}^l, \Phi(\boldsymbol{x}) \rangle + b^l \qquad \forall l.$$

The corresponding primal optimization problem is

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{l=1}^{K} \|\boldsymbol{w}^l\|_2^2 + C\sum_{i=1}^{N}\sum_{l=1}^{K} \xi_i^l \\ \text{with respect to} \quad & \boldsymbol{w}^l \in \mathbb{R}^S, \boldsymbol{\xi}^l \in \mathbb{R}_+^N, b^l \in \mathbb{R} \\ \text{subject to} \quad & (\langle \boldsymbol{w}^{y_i}, \Phi(\boldsymbol{x}_i) \rangle + b^{y_i}) - (\langle \boldsymbol{w}^l, \Phi(\boldsymbol{x}_i) \rangle + b^l) \geq 2 - \xi_i^l \qquad \forall (i, l \neq y_i) \\ & \xi_i^{y_i} = 0 \qquad \forall i. \end{aligned} \tag{1.9}$$

We obtain the Lagrangian dual of the primal problem (1.9) as follows:

$$\begin{aligned} L_D = & \frac{1}{2}\sum_{l=1}^{K} \|\boldsymbol{w}^l\|_2^2 + C\sum_{i=1}^{N}\sum_{l=1}^{K} \xi_i^l - \sum_{i=1}^{N}\sum_{l=1}^{K} \beta_i^l \xi_i^l \\ & - \sum_{i=1}^{N}\sum_{l=1}^{K} \alpha_i^l ((\langle \boldsymbol{w}^{y_i}, \Phi(\boldsymbol{x}_i) \rangle + b^{y_i}) - (\langle \boldsymbol{w}^l, \Phi(\boldsymbol{x}_i) \rangle + b^l) - 2 + \xi_i^l) \end{aligned}$$

and taking the derivatives of L_D with respect to the primal variables gives

$$\begin{aligned}
\frac{\partial L_D}{\partial \boldsymbol{w}^l} &= 0 \Rightarrow \boldsymbol{w}^l = \sum_{i=1}^{N} (\delta_{y_i}^l A_i - \alpha_i^l)\Phi(\boldsymbol{x}_i) \\
\frac{\partial L_D}{\partial b^l} &= 0 \Rightarrow \sum_{i=1}^{N} \alpha_i^l - \sum_{i=1}^{N} \delta_{y_i}^l A_i = 0 \qquad \forall l \\
\frac{\partial L_D}{\partial \xi_i^l} &= 0 \Rightarrow C = \alpha_i^l + \beta_i^l \qquad \forall (i,l)
\end{aligned} \tag{1.10}$$

where $\delta_{y_i}^l$ is 1 if $y_i = l$ and 0 otherwise, and $A_i = \sum_{l=1}^{K} \alpha_i^l$.

From L_D and (1.8), the dual formulation is obtained as

$$\begin{aligned}
\text{maximize} \quad & 2\sum_{i=1}^{N}\sum_{l=1}^{K} \alpha_i^l - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\delta_{y_i}^{y_j} A_i A_j - \sum_{l=1}^{K} \alpha_i^l (2\alpha_j^{y_i} - \alpha_j^l)\right) k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
\text{with respect to} \quad & \boldsymbol{\alpha}^l \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i^l - \sum_{i=1}^{N} \delta_{y_i}^l A_i = 0 \qquad \forall l \\
& (1 - \delta_{y_i}^l) C \geq \alpha_i^l \geq 0 \qquad \forall (i,l)
\end{aligned} \tag{1.11}$$

and the discriminant function for class l can be written as

$$f^l(\boldsymbol{x}) = \sum_{i=1}^{N} (\delta_{y_i}^l A_i - \alpha_i^l) k(\boldsymbol{x}_i, \boldsymbol{x}) + b^l.$$

The main disadvantage of this approach is the enormous size of the resulting quadratic programs. For example, one-versus-all method solves K separate N-variable quadratic problems, but the dual formulation in (1.11) has $N \times (K-1)$ variables. In order to tackle such large quadratic problems, different decomposition methods and optimization algorithms are proposed. Crammer and Singer (2001) develop a mathematical model that reduces the variable size from $N \times (K-1)$ to N and propose an efficient algorithm to solve the new formulation.

Figure 1.4 shows the discriminants and support vectors found by MCSVM with the linear kernel on a toy data set.

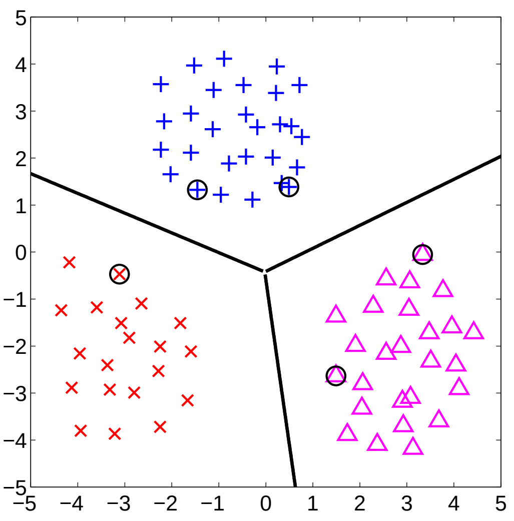


Figure 1.4. MCSVM solution on a three-class toy data set using k_L. The solid lines show the discriminants learned.

1.5. Posterior Probability Support Vector Machines

Tao *et al.* (2005b) in their proposed Posterior Probability SVM (PPSVM) modify the canonical SVM discussed above to utilize class probabilities instead of using hard $\{-1, +1\}$ labels. These "soft labels" are calculated from estimated posterior probabilities as

$$\hat{y}_i = 2\Pr(+1|\boldsymbol{x}_i) - 1 \qquad \forall i \tag{1.12}$$

and they rewrite the separation constraints as

$$\hat{y}_i(\langle \boldsymbol{w}, \Phi(\boldsymbol{x}_i)\rangle + b) \geq \hat{y}_i^2 - \hat{y}_i^2 \xi_i \qquad \forall i. \tag{1.13}$$

Note that the separation constraints can also be rewritten as (1.13) with hard labels, $\{y_i\}_{i=1}^N$ in place of soft labels, $\{\hat{y}_i\}_{i=1}^N$. In other words, y_i are equal to $\hat{y}_i$ when the posterior probability estimates in (1.12) are 0 or 1.

The advantage of using soft labels derived from posterior probabilities, $\{\hat{y}_i\}_{i=1}^N$, instead of hard class labels, $\{y_i\}_{i=1}^N$, in (1.13) is twofold. Because the posterior probability at a point is the combined effect of a number of neighboring instances, first, it gives a chance to correct the error introduced by wrongly labeled/noisy points due to correctly labeled neighbors; this can be seen as a smoothing of labels and therefore of the induced boundary. Second, an instance that is surrounded by a number of instances of the same class becomes redundant and this decreases the number of stored supports.

The multiclass extension to SVM, MCSVM, is modified to include posterior probability estimates instead of hard labels (Gönen *et al.*, 2008). The separation constraints in (1.9) try to place each instance on the correct side of each hyperplane with at least two units distance. In the canonical formulation, this comes from the fact that the true class label is $+1$ and the wrong class label is -1.

In the Multiclass PPSVM (MCPPSVM) formulation, the label of an instance for class l is defined as $2\Pr(l|\boldsymbol{x}) - 1$. So, the separation constraints in (1.9) should be replaced by the following constraints:

$$(\langle \boldsymbol{w}^{y_i}, \Phi(\boldsymbol{x}_i)\rangle + b^{y_i}) - (\langle \boldsymbol{w}^l, \Phi(\boldsymbol{x}_i)\rangle + b^l) \geq 2(\Pr(y_i|\boldsymbol{x}_i) - \Pr(l|\boldsymbol{x}_i)) - \xi_i^l \qquad \forall(i, l \neq y_i).$$

The objective function in (1.11) in the dual formulation becomes

$$\begin{aligned} \text{maximize} \quad & 2\sum_{i=1}^{N}\sum_{l=1}^{K}\alpha_i^l(\Pr(y_i|\boldsymbol{x}_i) - \Pr(l|\boldsymbol{x}_i)) \\ & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\delta_{y_i}^{y_j}A_iA_j - \sum_{l=1}^{K}\alpha_i^l(2\alpha_j^{y_i} - \alpha_j^l)\right)k(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned}$$

We show the difference between MCSVM and MCPPSVM on a three-class toy data set in Figure 1.5. We see that MCSVM stores the outliers as support vectors (filled points) and this shifts the induced boundary (solid lines). With MCPPSVM, because the neighbors of the outliers belong to a different class, the posterior probabilities there are small and the outliers are effectively canceled resulting in a reduction in bias; they are not chosen as support vectors and therefore they do not effect the boundary. Though this causes some training error, the generalization is improved, and the number of support vectors decreases from seven to four.

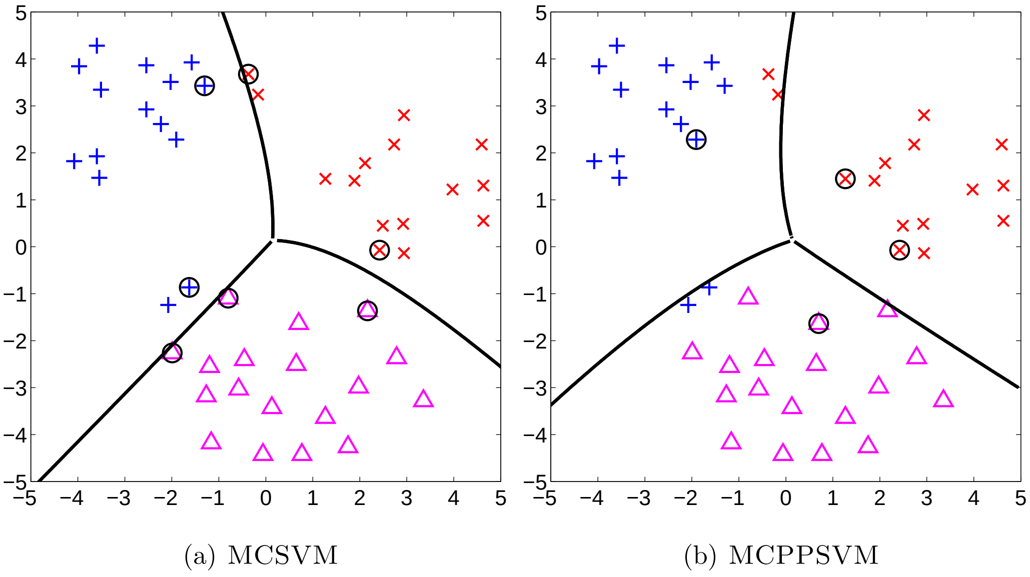


Figure 1.5. MCSVM and MCPPSVM solutions on a three-class toy data set using k_P $(q = 2)$.

1.6. Outline of the Thesis

The thesis is organized as follows: In Chapter 2, we review the multiple kernel learning methods and give two new formulations for regularizing the solutions and integrating the costs of kernels into the combination algorithm. The proposed localized multiple kernel learning method is introduced in Chapter 3. In Chapter 4, a supervised and localized dimensionality reduction algorithm based on kernel machines is developed. Empirical results obtained with the proposed methods of Chapters 2–4 and statistical comparisons with existing methods in the literature are presented in Chapter 5. We conclude and discuss possible future work in Chapter 6.

2. MULTIPLE KERNEL LEARNING

Kernel machines learn a decision function in terms of kernel values between training instances, $\{\boldsymbol{x}_i\}_{i=1}^N$, and the test instance, $\boldsymbol{x}$, as follows:

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

where the kernel function $k(\cdot, \cdot)$ can be one of the general purpose kernels (e.g., linear, polynomial, and Gaussian) or one that is application-specific (e.g., a graph kernel, a tree kernel, and so on). Selecting the kernel function $k(\cdot, \cdot)$ and its parameters (e.g., degree in the polynomial kernel or spread in the Gaussian kernel) is an important issue in training. Generally, a cross-validation procedure is used to choose the best performing kernel function among a set of kernel functions on a separate validation set different from the training set.

In recent years, Multiple Kernel Learning (MKL) methods have been proposed, where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters:

$$k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = f_{\eta}(\{k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)\}_{m=1}^P) \tag{2.1}$$

where the combination function, $f_{\eta}(\cdot)$, can be a linear or a nonlinear function of the input kernels. Kernel functions, $\{k_m(\cdot, \cdot)\}_{m=1}^P$, take P feature representations (not necessarily different) of data instances, where $\boldsymbol{x}_i = \{\boldsymbol{x}_i^m\}_{m=1}^P$, $\boldsymbol{x}_i^m \in \mathbb{R}^{D_m}$, and D_m is the dimensionality of the corresponding feature representation.

The reasoning is similar to combining different classifiers: Instead of choosing a single kernel function and putting all our eggs in the same basket, it is better to have a set and let an algorithm do the picking or combination. There can be two uses of MKL:

(a) Different kernels correspond to different notions of similarity and instead of trying to find which works best, a learning method does the picking for us, or may use a combination of them. Using a specific kernel may be a source of bias, and in allowing a learner to choose among a set of kernels, a better solution can be found.

(b) Different kernels may be using inputs coming from different representations possibly from different sources or modalities. Since these are different representations, they have different measures of similarity corresponding to different kernels. In such a case, combining kernels is one possible way to combine multiple information sources. Noble (2004) call this method of combining kernels *intermediate combination* and contrasts this with *early combination* (where features from different sources are concatenated and fed to a single learner) and *late combination* (where different features are fed to different classifiers whose decisions are then combined by a fixed or trained combiner).

There is significant amount of work in the machine learning literature for combining multiple kernels. We give a taxonomy in Table 2.1 and the following sections follow this taxonomy (Gönen and Alpaydın, 2009b).

2.1. Fixed Rules for Kernel Combination

Fixed rules for kernel combination use $f_\eta(\cdot)$ in (2.1) as a fixed function of the kernels, without any training. Once we obtain $k_\eta(\cdot,\cdot)$ using $f_\eta(\cdot)$, we train a canonical kernel machine with the kernel matrix calculated using $k_\eta(\cdot,\cdot)$. For example, we can obtain a valid kernel by taking the summation or multiplication of two valid kernels (Cristianini and Shawe-Taylor, 2000):

$$\begin{aligned} k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1) + k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2) \\ k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1) k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2). \end{aligned} \tag{2.2}$$

Table 2.1. Taxonomy of multiple kernel learning algorithms.

FIXED RULES
Pavlidis *et al.* (2001); Ben-Hur and Noble (2005)
PARAMETERIZED FUNCTIONS
Linear Functions
Linear Combination
Lanckriet *et al.* (2002); Lanckriet *et al.* (2004a); Conforti and Guido (2010)
Nonnegative Linear Combination
Fung *et al.* (2004); Lanckriet *et al.* (2004a); Tsuda *et al.* (2004); Qiu and Lane (2005); Lin *et al.* (2009); Zhao *et al.* (2009); Kloft *et al.* (2010)
Convex Combination
Bousquet and Herrmann (2003); Bach *et al.* (2004); Argyriou *et al.* (2005); Argyriou *et al.* (2006); Micchelli and Pontil (2005); Kim *et al.* (2006); Sonnenburg *et al.* (2006a); Sonnenburg *et al.* (2006b); De Bie *et al.* (2007); Rakotomamonjy *et al.* (2007); Rakotomamonjy *et al.* (2008); Ye *et al.* (2007a); Zien and Ong (2007); Longworth and Gales (2008); Longworth and Gales (2009); Xu *et al.* (2009a)
Binary Combination
Xu *et al.* (2009b)
Nonlinear Functions
Ong *et al.* (2003); Ong *et al.* (2005); Ong and Smola (2003); Lewis *et al.* (2006a); Varma and Ray (2007); Varma and Babu (2009); Cortes *et al.* (2010)
SIMILARITY-BASED METHODS
Kernel Alignment
Linear Combination
Lanckriet *et al.* (2004a); Igel *et al.* (2007)
Nonnegative Linear Combination
Kandola *et al.* (2002); Lanckriet *et al.* (2004a)
Convex Combination
Qiu and Lane (2009)
Other Similarity Measures
He *et al.* (2008); Nguyen and Ho (2008); Ying *et al.* (2009)
BOOSTING METHODS
Bennett *et al.* (2002); Crammer *et al.* (2003); Bi *et al.* (2004)
BAYESIAN METHODS
Girolami and Rogers (2005); Girolami and Zhong (2007); Damoulas and Girolami (2008); Damoulas and Girolami (2009a); Damoulas and Girolami (2009b)

We can apply the rules in (2.2) recursively to obtain the rules for more than two kernels:

$$\begin{aligned} k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{m=1}^{P} k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) \\ k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \prod_{m=1}^{P} k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m). \end{aligned} \tag{2.3}$$

Pavlidis *et al.* (2001) report that on a gene functional classification task, training an SVM with an unweighted sum of heterogeneous kernels gives better results than the combination of multiple SVMs each trained with one of these kernels.

Pairwise kernels are used to express the similarity between pairs (e.g., of proteins in terms of similarities between individual proteins). The simplest way to define a pairwise kernel is

$$k(\{\boldsymbol{x}_i^a, \boldsymbol{x}_j^a\}, \{\boldsymbol{x}_i^b, \boldsymbol{x}_j^b\}) = k(\boldsymbol{x}_i^a, \boldsymbol{x}_i^b)k(\boldsymbol{x}_j^a, \boldsymbol{x}_j^b) + k(\boldsymbol{x}_i^a, \boldsymbol{x}_j^b)k(\boldsymbol{x}_j^a, \boldsymbol{x}_i^b).$$

Ben-Hur and Noble (2005) combine pairwise kernels in two different ways

$$\begin{aligned} k_\eta(\{\boldsymbol{x}_i^a, \boldsymbol{x}_j^a\}, \{\boldsymbol{x}_i^b, \boldsymbol{x}_j^b\}) &= \sum_{m=1}^{P} k_m(\{\boldsymbol{x}_i^a, \boldsymbol{x}_j^a\}, \{\boldsymbol{x}_i^b, \boldsymbol{x}_j^b\}) \\ k_\eta(\{\boldsymbol{x}_i^a, \boldsymbol{x}_j^a\}, \{\boldsymbol{x}_i^b, \boldsymbol{x}_j^b\}) &= k_\eta(\boldsymbol{x}_i^a, \boldsymbol{x}_i^b)k_\eta(\boldsymbol{x}_j^a, \boldsymbol{x}_j^b) + k_\eta(\boldsymbol{x}_i^a, \boldsymbol{x}_j^b)k_\eta(\boldsymbol{x}_j^a, \boldsymbol{x}_i^b) \end{aligned}$$

and improve the classification performance for protein-protein interaction prediction task.

2.2. Learning Parameterized Functions of Kernels

In (2.1), instead of using a fixed $f_\eta(\cdot)$, we can have a function parameterized by a set of parameters $\boldsymbol{\Theta}$ and then we have a learning procedure to optimize $\boldsymbol{\Theta}$ as well.

2.2.1. Linear Functions of Kernels

The simplest case is to parameterize the sum rule using kernel weights, $\{\eta_m\}_{m=1}^P$:

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = f_\eta(\{k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)\}_{m=1}^P | \boldsymbol{\Theta} = \{\eta_m\}_{m=1}^P) = \sum_{m=1}^P \eta_m k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m).$$

Different versions of this approach differ in the way they put restrictions on $\{\eta_m\}_{m=1}^P$.

<u>2.2.1.1. Linear Combination.</u> A direct approach to optimize the unrestricted kernel combination weights can be followed (Lanckriet *et al.*, 2002; Lanckriet *et al.*, 2004a). The implausibility of a kernel matrix, $\omega(\mathbf{K})$, is defined as the objective function value obtained after solving a canonical SVM optimization problem (Here we only consider the soft margin formulation, which uses the l_1-norm on slack variables).

$$\begin{aligned} \text{maximize} \quad & \omega(\mathbf{K}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to} \quad & \sum_{i=1}^N \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \qquad \forall i \end{aligned}$$

The combined kernel matrix is selected from the following set:

$$\mathcal{K}_L = \left\{ \mathbf{K} : \mathbf{K} = \sum_{m=1}^P \eta_m \mathbf{K}_m, \;\; \mathbf{K} \succeq 0, \;\; \operatorname{tr}(\mathbf{K}) \leq c \right\}$$

where the selected kernel matrix is forced to be positive semidefinite.

The resulting optimization problem that minimizes the implausibility of the combined kernel matrix (the objective function value of the corresponding soft margin SVM optimization problem) is formulated as

$$\begin{aligned} \text{minimize} \quad & \omega(\mathbf{K}_\eta^{\text{tra}}) \\ \text{with respect to} \quad & \mathbf{K}_\eta \in \mathcal{K}_L \\ \text{subject to} \quad & \text{tr}(\mathbf{K}_\eta) = c \end{aligned}$$

where $\mathbf{K}_\eta^{\text{tra}}$ is the kernel matrix calculated over only the training set and this problem can be cast into the following Semidefinite Programming (SDP) formulation:

$$\begin{aligned} \text{minimize} \quad & t \\ \text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}^P, t \in \mathbb{R}, \lambda \in \mathbb{R}, \boldsymbol{\nu} \in \mathbb{R}_+^N, \boldsymbol{\delta} \in \mathbb{R}_+^N \\ \text{subject to} \quad & \text{tr}(\mathbf{K}_\eta) = c \\ & \begin{pmatrix} (\boldsymbol{y}\boldsymbol{y}^\top)\mathbf{K}_\eta^{\text{tra}} & \boldsymbol{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda \boldsymbol{y} \\ (\boldsymbol{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda \boldsymbol{y})^\top & t - 2C\boldsymbol{\delta}^\top \boldsymbol{e} \end{pmatrix} \succeq 0 \\ & \mathbf{K}_\eta \succeq 0. \end{aligned}$$

This optimization problem is defined for a transductive learning setting and we need to be able to calculate the kernel function values for test instances as well as training instances.

Different kernels calculated on heterogeneous genomic data, namely, amino acid sequences, protein-protein interactions, genetic interactions, protein complex data, and expression data, are combined using SDP formulation for predicting function classifications associated with Yeast proteins (Lanckriet *et al.*, 2004c; Lanckriet *et al.*, 2004a). This gives better results than SVMs trained with each kernel in nine out of 13 experiments. Conforti and Guido (2010) propose another SDP formulation that removes the trace restriction on the combined kernel matrix and introduces constraints over the kernel weights for an inductive setting.

<u>2.2.1.2. Nonnegative Linear Combination.</u> Lanckriet *et al.* (2004a) restrict the combination weights to have nonnegative values by selecting the combined kernel matrix from

$$\mathcal{K}_P = \left\{ \mathbf{K} : \mathbf{K} = \sum_{m=1}^{P} \eta_m \mathbf{K}_m, \;\; \boldsymbol{\eta} \geq 0, \;\; \mathbf{K} \succeq 0, \;\; \operatorname{tr}(\mathbf{K}) \leq c \right\}.$$

and reduce the SDP formulation to the following Quadratically Constrained QP (QCQP) optimization problem by selecting the combined kernel matrix from $\mathcal{K}_P$:

$$\begin{aligned} \text{minimize} \;\; & \frac{1}{2} ct - \sum_{i=1}^{N} \alpha_i \\ \text{with respect to} \;\; & \boldsymbol{\alpha} \in \mathbb{R}_+^N, t \in \mathbb{R} \\ \text{subject to} \;\; & \operatorname{tr}(\mathbf{K}_m) t \geq \boldsymbol{\alpha}^\top \left((\boldsymbol{y}\boldsymbol{y}^\top) \mathbf{K}_m^{\text{tra}} \right) \boldsymbol{\alpha} \qquad \forall m \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \qquad \forall i \end{aligned}$$

where we can jointly find the support vector coefficients and the kernel combination weights. This optimization problem is also developed for a transductive setting, but we can simply take the number of test instances as zero and find the kernel combination weights for an inductive setting. The interior-point methods used to solve this QCQP formulation also return the optimal values of the dual variables, and they correspond to the optimal kernel weights. Qiu and Lane (2005) give Semi-Infinite Linear Programming (SILP) and QCQP formulations of regression estimation using ϵ-tube SVR. The QCQP formulation is used for predicting siRNA efficacy by combining kernels over heterogeneous data sources (Qiu and Lane, 2009). Zhao *et al.* (2009) develop a multiple kernel learning method for clustering problems using the maximum margin clustering idea of Xu *et al.* (2005) and a nonnegative linear combination of kernels.

Lanckriet *et al.* (2004a) combine two different kernels obtained from heterogeneous information sources, namely, bag-of-words and graphical representations, on

Reuters-21578 data set. Combining these two kernels with positive weights outperforms single-kernel results obtained with SVM on four tasks out of five. Lanckriet *et al.* (2004b) uses a QCQP formulation to integrate multiple kernel functions calculated on heterogeneous views of genome data obtained through different experimental procedures. These views include amino acid sequences, hydropathy profiles, gene expression data and known protein-protein interactions. The prediction task is to recognize the particular classes of proteins, namely, membrane proteins and ribosomal proteins. The QCQP approach gives significantly better results than any single kernel and the unweighted sum of kernels. The assigned kernel weights also enable us to extract the relative importance of the data sources feeding the separate kernels. This approach assigns near zero weights to random kernels added to the candidate set of kernels before training. Dehak *et al.* (2008) combine three different kernels obtained on the same features and get better results than score fusion for speaker verification tasks.

Fung *et al.* (2004) propose an iterative algorithm based on kernel Fisher discriminant analysis to combine heterogeneous kernels in a linear manner with nonnegative weights. The proposed method requires solving a simple nonsingular system of linear equations of size $(N + 1)$ and a QP problem having P decision variables at each iteration. On a colorectal cancer diagnosis task, this method obtains similar results using much less computation time compared to selecting a kernel for standard kernel Fisher discriminant analysis.

Tsuda *et al.* (2004) learn the kernel combination weights by minimizing an approximation of the cross-validation error for kernel Fisher discriminant analysis. In order to update the kernel combination weights, cross-validation error should be approximated with a differentiable error function. Tsuda *et al.* (2004) use the sigmoid function for approximation and derive the update rules of the kernel weights. This procedure requires inverting a $N \times N$ matrix and calculating the gradients at each step. Tsuda *et al.* (2004) combine heterogeneous data sources using kernels, which are mixed linearly and nonlinearly, for bacteria classification and gene function prediction tasks. Fisher discriminant analysis with the combined kernel matrix, which is optimized using the cross-validation error approximation, gives significantly better results than single kernels for both tasks.

In order to consider the capacity of the resulting classifier, Tan and Wang (2004) optimize the nonnegative combination coefficients by using the minimal upper bound of the Vapnik-Chervonenkis dimension as the target function.

Lin *et al.* (2009) propose a dimensionality reduction method that uses multiple kernels to embed data instances from different feature spaces to a unified feature space. The method is derived from a graph embedding framework using kernel matrices instead of data matrices. The learning phase is performed using a two-step alternating optimization procedure that updates the dimensionality reduction coefficients and the kernel weights in turn. McFee and Lanckriet (2009) propose a method for learning a unified space from multiple kernels calculated over heterogeneous data sources. This method uses a partial order over pairwise distances as the input and produces an embedding using graph-theoretic tools. The kernel (data source) combination rule is learned by solving an SDP problem and all input instances are mapped to the constructed common embedding space.

Kloft *et al.* (2010) generalize the MKL formulation for arbitrary l_p ($p \geq 1$) norms by regularizing over the kernel coefficients (which is done by adding $\mu\|\boldsymbol{\eta}\|_p^p$ to the objective function) or equivalently constraining them ($\|\boldsymbol{\eta}\|_p^p \leq 1$). The resulting optimization problem is

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\left(\sum_{m=1}^{P}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_i y_i y_j k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)\right)^{\frac{p-1}{p}}\right)^{\frac{p}{p-1}} \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i.
\end{aligned}$$

and Kloft *et al.* (2010) solve this problem by using alternating optimization strategies based on Newton descent and cutting planes.

Lewis *et al.* (2006b) compare the performances of unweighted and weighted sums of kernels on a gene functional classification task. Their results can be summarized with two guidelines:

(a) When all kernels or data sources are informative, we should use the unweighted sum rule.
(b) When some of the kernels or the data sources are noisy or irrelevant, we should optimize the kernel weights.

Usually, the kernel weights are constrained by a trace or l_1-norm regularization. Cortes *et al.* (2009) discuss the suitability of the l_2-norm for MKL. They combine kernels with ridge regression using the l_2-norm regularization over the kernel weights. They conclude that using the l_1-norm improves the performance for a small number of kernels, but degrades the performance when combining a large number of kernels. However, the l_2-norm never decreases the performance and significantly increases the performance for larger sets of candidate kernels. Yan *et al.* (2009) compare the l_1-norm and the l_2-norm for image and video classification tasks, and conclude that the l_2-norm should be used when the combined kernels carry complementary information.

2.2.1.3. Convex Combination. We can think of kernel combination as a weighted average of kernels and consider $\boldsymbol{\eta} \in \mathbb{R}_+^P$ and $\sum_{m=1}^{P} \eta_m = 1$. Joachims *et al.* (2001) show that combining two kernels is beneficial if both of them achieve approximately the same performance and use different data instances as support vectors. This makes sense: In combination, we want kernels to be useful by themselves and complementary. In a web page classification experiment, combining the word and the hyperlink representations through the convex combination of two kernels (i.e., $\eta_2 = 1 - \eta_1$) can achieve better classification accuracy than each of the kernels.

Chapelle *et al.* (2002) calculate the derivative of the margin and the derivative of the radius (of the smallest sphere enclosing the training points) with respect to a kernel parameter, θ:

$$\begin{aligned}
\frac{\partial \|\boldsymbol{w}\|_2^2}{\partial \theta} &= -\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \theta} \\
\frac{\partial R^2}{\partial \theta} &= \sum_{i=1}^{N} \beta_i \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_i)}{\partial \theta} - \sum_{i=1}^{N}\sum_{j=1}^{N} \beta_i \beta_j \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \theta}
\end{aligned}$$

where $\boldsymbol{\alpha}$ is obtained by solving the canonical SVM optimization problem and $\boldsymbol{\beta}$ is obtained by solving the QP problem defined by Vapnik (1998). These derivatives can be used to optimize the individual parameters (e.g., scaling coefficient) on each feature by using an alternating optimization procedure (Weston *et al.*, 2001; Chapelle *et al.*, 2002; Grandvalet and Canu, 2003). This strategy is also a multiple kernel learning approach because the optimized parameters can be interpreted as the kernel parameters, and then we combine these kernel values over all features.

Bousquet and Herrmann (2003) rewrite the gradient of the margin by replacing $\mathbf{K}$ with $\mathbf{K}_{\boldsymbol{\eta}}$ and taking the derivative with respect to the kernel weights gives

$$\frac{\partial \|\boldsymbol{w}_{\boldsymbol{\eta}}\|_2^2}{\partial \eta_m} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k_{\boldsymbol{\eta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \eta_m} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) \qquad \forall m$$

where $\boldsymbol{w}_{\boldsymbol{\eta}}$ is the weight vector obtained using $\mathbf{K}_{\boldsymbol{\eta}}$ in training. In an iterative manner, an SVM is trained to obtain $\boldsymbol{\alpha}$, then $\boldsymbol{\eta}$ is updated using the calculated gradient while considering nonnegativity (i.e., $\boldsymbol{\eta} \in \mathbb{R}_+^P$) and normalization (i.e., $\sum_{m=1}^{P} \eta_m = 1$). This procedure considers the performance (in terms of margin maximization) of the resulting classifier, which uses the combined kernel matrix.

Micchelli and Pontil (2005) try to learn an optimal kernel over the convex hull of predefined basic kernels by minimizing a regularization functional. Their analysis shows that any optimizing kernel can be expressed as the convex combination of basic kernels. Practical algorithms for learning a suboptimal kernel when the basic kernels are continuously parameterized by a compact set can be built (Argyriou *et al.*, 2005; Argyriou *et al.*, 2006). This continuous parameterization allows us to select kernels from basically an infinite set instead of a finite number of basic kernels.

Kim *et al.* (2006) show that selecting the optimal kernel from the set of convex combinations over the candidate kernels can be formulated as a convex optimization problem. This formulation is more efficient than the iterative approach of Fung *et al.* (2004). Ye *et al.* (2007a) formulate an SDP problem inspired by Kim *et al.* (2006) for learning an optimal kernel over a convex set of candidate kernels for regularized kernel discriminant analysis. The SDP formulation can be modified so that it can jointly optimize the kernel weights and the regularization parameter. The QCQP and SILP formulations equivalent to the previous SDP problem are derived in order to reduce the time complexity (Ye *et al.*, 2007b; Ye *et al.*, 2008). These three formulations are directly applicable to multiclass classification due to being dependent on regularized kernel discriminant analysis.

De Bie *et al.* (2007) derive a QCQP formulation of one-class classification problem using a convex combination of multiple kernels. In order to prevent the combined kernel from overfitting, they also propose a modified mathematical model that puts lower limits for the kernel weights. Hence, each kernel in the set of candidate kernels is used in the combined kernel and we obtain a more regularized learner. Their results on gene prioritization task coincide with that of Lewis *et al.* (2006b).

Bach *et al.* (2004) propose a modified primal formulation that uses the weighted l_1-norm on feature spaces and the l_2-norm within each feature space. The modified primal formulation is

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\left(\sum_{m=1}^{P} d_m \|\boldsymbol{w}_m\|_2\right)^2 + C\sum_{i=1}^{N}\xi_i \\ \text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{S_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to} \quad & y_i\left(\sum_{m=1}^{P}\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i^m)\rangle + b\right) \geq 1 - \xi_i \qquad \forall i \end{aligned} \tag{2.4}$$

where the feature space constructed by using $\Phi_m(\cdot)$ has the dimensionality S_m and the weight d_m. When we consider this optimization problem as a SECOND-ORDER CONE PROGRAMMING (SOCP), we obtain the following dual formulation:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\gamma^2 - \sum_{i=1}^{N} \alpha_i \\
\text{with respect to} \quad & \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \gamma^2 d_m^2 \geq \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) \qquad \forall m \\
& \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i
\end{aligned} \tag{2.5}$$

where we again get the optimal kernel weights from the optimal dual variables and the weights satisfy $\sum_{m=1}^{P} d_m^2 \eta_m = 1$. The dual problem is exactly equivalent to the QCQP formulation of Lanckriet *et al.* (2004a) when we take $d_m = \sqrt{\text{tr}(\mathbf{K}_m)/c}$. The advantage of the SOCP formulation is that Bach *et al.* (2004) devise an SMO-like algorithm by adding a Moreau-Yosida regularization term, $1/2 \sum_{m=1}^{P} a_m^2 \|\boldsymbol{w}_m\|_2^2$, to the primal objective function and deriving the corresponding dual formulation. Using the l_1-norm on feature spaces, Yamanishi *et al.* (2007) combine tree kernels for identifying human glycans into four blood components: leukemia cells, erythrocytes, plasma, and serum. Except on plasma task, representing glycans as rooted trees and combining kernels improve performance in terms of area under the curve. Bach (2009) develops a method for learning linear combinations of an exponential number of kernels, which can be expressed as product of sums. The method is applied to nonlinear variable selection and efficiently explores the large feature spaces in polynomial time.

The QCQP formulation of Bach *et al.* (2004) can be rewritten as

$$\begin{aligned}
\text{minimize} \quad & \gamma \\
\text{with respect to} \quad & \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i \\
& \gamma \geq \underbrace{\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) - \sum_{i=1}^{N} \alpha_i}_{\mathcal{S}_m(\boldsymbol{\alpha})} \qquad \forall m
\end{aligned}$$

and can be converted into the following SILP problem:

$$\begin{aligned}
\text{maximize} \quad & \theta \\
\text{with respect to} \quad & \theta \in \mathbb{R}, \boldsymbol{\eta} \in \mathbb{R}_+^P \\
\text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1 \\
& \sum_{m=1}^{P} \eta_m \mathcal{S}_m(\boldsymbol{\alpha}) \geq \theta \qquad \forall \boldsymbol{\alpha} \in \{\boldsymbol{\alpha} : \boldsymbol{\alpha} \in \mathbb{R}^N, \;\; \boldsymbol{\alpha}^\top \boldsymbol{y} = 0, \;\; C \geq \boldsymbol{\alpha} \geq 0\}
\end{aligned}$$

where the problem has infinitely many constraints due to possible values of $\boldsymbol{\alpha}$ (Sonnenburg *et al.*, 2006a; Sonnenburg *et al.*, 2006b).

The SILP formulation has lower computational complexity compared to the SDP and QCQP formulations. A column generation approach is used to solve the resulting SILPs using a generic Linear Programming (LP) solver and a canonical SVM solver in the inner loop (Sonnenburg *et al.*, 2006a; Sonnenburg *et al.*, 2006b). Both the LP solver and the SVM solver can use previous optimal values for hot-start to obtain the new optimal values faster. These allow us to use the SILP formulation to learn the kernel combination weights for hundreds of kernels on hundreds of thousands of training instances efficiently. For example, they perform training on a million real-world splice data set from computational biology with string kernels. They also generalize the idea to regression estimation, one-class classification, and strictly convex and differentiable loss functions.

Zien and Ong (2007) develop a QCQP formulation and convert this formulation in two different SILP problems for multiclass classification problems. They show that their formulation is the multiclass generalization of the previously developed binary classification methods of Bach *et al.* (2004) and Sonnenburg *et al.* (2006b). The proposed multiclass formulation is tested on different bioinformatics applications such as bacterial protein location prediction (Zien and Ong, 2007) and protein subcellular location prediction (Zien and Ong, 2007; Zien and Ong, 2008), and outperforms individual kernels and unweighted sum of kernels. Hu *et al.* (2009) combine the generalized multiple kernel learning formulation of Zien and Ong (2007) and the sparse kernel learning method of Wu *et al.* (2006). This hybrid approach learns the optimal kernel weights and also obtains a sparse solution.

A different primal problem for MKL and a projected gradient method to solve this optimization problem are proposed (Rakotomamonjy *et al.*, 2007; Rakotomamonjy *et al.*, 2008). The proposed primal formulation is

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{m=1}^{P}\frac{1}{\eta_m}\|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i \\ \text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{S_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R}, \boldsymbol{\eta} \in \mathbb{R}_+^P \\ \text{subject to} \quad & y_i\left(\sum_{m=1}^{P}\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i)\rangle + b\right) \geq 1 - \xi_i \qquad \forall i \\ & \sum_{m=1}^{P}\eta_m = 1 \end{aligned}$$

and the optimal SVM objective function value given $\boldsymbol{\eta}$, $J(\boldsymbol{\eta})$, is defined as

$$\begin{aligned}
\text{minimize} \quad & J(\boldsymbol{\eta}) = \frac{1}{2}\sum_{m=1}^{P} \frac{1}{\eta_m}\|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i \\
\text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{S_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\
\text{subject to} \quad & y_i\left(\sum_{m=1}^{P}\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i)\rangle + b\right) \geq 1 - \xi_i \qquad \forall i.
\end{aligned}$$

Due to strong duality, one can also calculate $J(\boldsymbol{\eta})$ using the dual formulation:

$$\begin{aligned}
\text{maximize} \quad & J(\boldsymbol{\eta}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i.
\end{aligned}$$

The primal formulation can be seen as the following constrained optimization problem:

$$\begin{aligned}
\text{minimize} \quad & J(\boldsymbol{\eta}) \\
\text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}_+^P \\
\text{subject to} \quad & \sum_{m=1}^{P}\eta_m = 1.
\end{aligned}$$

The overall procedure to solve this problem, called SimpleMKL, consists of two main steps: (a) solving a canonical SVM optimization problem with given $\boldsymbol{\eta}$, and, (b) updating $\boldsymbol{\eta}$ using the following gradient calculated with $\boldsymbol{\alpha}$ found in the first step:

$$\frac{\partial J(\boldsymbol{\eta})}{\partial \eta_m} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) \qquad \forall m.$$

The gradient update procedure must consider the nonnegativity and normalization properties of the kernel weights. The derivative with respect to the kernel weights is exactly equivalent (up to a multiplicative constant) to the gradient of the margin calculated by Bousquet and Herrmann (2003). The overall algorithm is very similar to the algorithm used by Sonnenburg *et al.* (2006a) and Sonnenburg *et al.* (2006b) to solve SILP formulation. Both algorithms use a canonical SVM solver in order to calculate $\boldsymbol{\alpha}$ at each step. The difference is that they use different updating procedures for $\boldsymbol{\eta}$, namely, a projected gradient update and solving an LP. SimpleMKL is more stable than solving the SILP formulation and can be generalized to regression estimation, one-class classification, and multiclass classification (Rakotomamonjy *et al.*, 2008).

Xu *et al.* (2009a) propose a hybrid method that combines the SILP formulation (Sonnenburg *et al.*, 2006b) and SimpleMKL (Rakotomamonjy *et al.*, 2008). The SILP formulation does not regularize the kernel weights obtained from the cutting plane method and SimpleMKL uses the gradient calculated only in the last iteration. The proposed model overcomes both disadvantages and finds the kernel weights for the next iteration by solving a small QP problem; this regularizes the solution and uses the past information.

To the objective function of SimpleMKL (Rakotomamonjy *et al.*, 2008), an extra regularization term can also be introduced (Longworth and Gales, 2008; Longworth and Gales, 2009). This allows us to change the level of sparsity for the kernels used in the combined kernel. The extra regularization term is

$$\lambda \sum_{m=1}^{P} \left(\eta_m - \frac{1}{P} \right)^2 = \lambda \left(\sum_{m=1}^{P} \eta_m^2 - \frac{1}{P} \right) =^{+} \lambda \sum_{m=1}^{P} \eta_m^2$$

where λ is regularization parameter that determines the solution sparsity. For example, large values of λ force the mathematical model to use all kernels with uniform weight, whereas small values produce sparse combinations.

Instead of selecting kernels from a predefined finite set, we can increase the number of candidate kernels in an iterative manner. Gehler and Nowozin (2008) propose a new algorithm that can basically select kernels from an uncountable infinite set. Their forward selection algorithm finds the kernel weights for a fixed size of candidate kernels by using one of the methods described above, then adds a new kernel to the set of candidate kernels, until convergence.

Most MKL methods do not consider the group structure between the kernels combined. For example, a group of kernels may be calculated on the same set of features and even if we assign a nonzero weight to only one of them, we have to extract the features in the testing phase. When kernels have such a group structure, it is reasonable to pick all or none of them in the combined kernel. Szafranski *et al.* (2008) follow this idea and derive a MKL method by changing the mathematical model used by Rakotomamonjy *et al.* (2007).

Subrahmanya and Shin (2010) generalize group-feature selection to kernel selection by introducing a log-based concave penalty term for obtaining extra sparsity; this is called Sparse MKL (SMKL). The reason for adding this concave penalty term is explained as the lack of ability of convex MKL methods to obtain sparse formulations. They show that SMKL obtains more sparse solutions than convex formulations for signal processing applications.

Tanabe *et al.* (2008) propose the following rule in order to choose the kernel weights for classification problems:

$$\eta_m = \frac{\pi_m - \delta}{\sum\limits_{h=1}^{P} (\pi_h - \delta)}$$

where π_m is the accuracy obtained using only $\mathbf{K}_m$ and δ is the threshold that should be less than or equal to the minimum of the accuracies obtained from single-kernel learners. Qiu and Lane (2009) propose two simple heuristics to select the kernel weights for regression estimation problems:

$$\eta_m = \frac{R_m}{\sum\limits_{h=1}^{P} R_h} \qquad \forall m$$

$$\eta_m = \frac{\sum\limits_{h=1}^{P} M_h - M_m}{(P-1)\sum\limits_{h=1}^{P} M_h} \qquad \forall m$$

where R_m is the Pearson correlation coefficient the predicted labels generated by the regressor using kernel matrix $\mathbf{K}_m$ and the true labels, and M_m is the mean square error generated by the regressor using kernel matrix $\mathbf{K}_m$.

2.2.1.4. Binary Combination. Another possibility is to allow only binary η_m for kernel selection. We get rid of kernels whose $\eta_m = 0$ and use the kernels whose $\eta_m = 1$.

Xu *et al.* (2009b) define a combined kernel over the set of kernels calculated on each feature independently and perform feature selection using this definition. The defined kernel function can be expressed as

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{D} \eta_m k(\boldsymbol{x}_i[m], \boldsymbol{x}_j[m])$$

where $\boldsymbol{\eta} \in \{0,1\}^D$. For efficient learning, $\boldsymbol{\eta}$ is relaxed into the continuous domain (i.e., $1 \geq \boldsymbol{\eta} \geq 0$). Following Lanckriet *et al.* (2004a), an SDP formulation is derived and this formulation is cast into a QCQP problem in order to reduce the time complexity.

2.2.2. Nonlinear Functions of Kernels

A linear combination may be restrictive and other combinations are also possible.

Ong *et al.* (2003) propose learning a kernel function instead of a kernel matrix. They define a kernel function in the space of kernels called a *hyperkernel*. Their construction includes convex combinations of an infinite number of pointwise non-negative kernels. Hyperkernels are generalized to different machine learning problems such as classification, regression estimation, and one-class classification (Ong and Smola, 2003; Ong *et al.*, 2005). When they use the regularized risk functional as the empirical quality functional to be optimized, the learning phase can be performed by solving an SDP problem. Tsang and Kwok (2006) convert the resulting optimization problems into SOCP problems in order to reduce the time complexity of the training phase.

de Diego *et al.* (2004) define a functional form of combining two kernels:

$$\mathbf{K}\eta = \frac{1}{2}(\mathbf{K}_1 + \mathbf{K}_2) + f(\mathbf{K}_1 - \mathbf{K}_2)$$

where the term $f(\mathbf{K}_1 - \mathbf{K}_2)$ represents the difference of information between what $\mathbf{K}_1$ and $\mathbf{K}_2$ provide for classification. They investigate three different functions:

$$\begin{aligned}
k\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \frac{1}{2}(k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1) + k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2)) + \tau y_i y_j |k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1) - k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2)| \\
k\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \frac{1}{2}(k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1) + k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2)) + \tau y_i y_j (k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1) - k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2)) \\
\mathbf{K}\eta &= \frac{1}{2}(\mathbf{K}_1 + \mathbf{K}_2) + \tau(\mathbf{K}_1 - \mathbf{K}_2)(\mathbf{K}_1 - \mathbf{K}_2)
\end{aligned}$$

where $\tau \in \mathbb{R}_+$ is the parameter that represents the weight assigned to the term $f(\mathbf{K}_1 - \mathbf{K}_2)$ selected through cross-validation procedure and the first two functions do not ensure having positive semidefinite kernel matrices. More than two kernel functions can also be combined by applying these rules recursively.

Moguerza *et al.* (2004) propose a matrix functional form of combining kernels:

$$k\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i, \boldsymbol{x}_j) k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)$$

where $\eta_m(\cdot, \cdot)$ assigns a weight to $k_m(\cdot, \cdot)$ according to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. They propose different heuristics to estimate the weighing functions.

Lee *et al.* (2007) combine kernels using a compositional method that constructs a $(P \times N) \times (P \times N)$ compositional kernel matrix. This matrix and P times replicated training instances are used to train a canonical SVM. Lewis *et al.* (2006a) use a latent variable generative model using the maximum entropy discrimination to learn data-dependent kernel combination weights. This method combines a generative probabilistic model with a discriminative large margin method.

A generalized formulation of MKL called Generalized MKL (GMKL) that contains two regularization terms and a loss function in the objective function is proposed (Varma and Ray, 2007; Varma and Babu, 2009). This formulation tries to regularize both the hyperplane weight and the kernel combination weights. The loss function can be one of the classical loss functions, such as hinge loss for classification or ϵ-loss for regression estimation. The proposed primal formulation applied to binary classification problem with hinge loss and the regularization function $r(\cdot)$ can be written as

$$\begin{aligned} \text{minimize} \;\; & \frac{1}{2}\|\boldsymbol{w}_{\eta}\|_2^2 + C\sum_{i=1}^{N}\xi_i + r(\boldsymbol{\eta}) \\ \text{with respect to} \;\; & \boldsymbol{w}_{\eta} \in \mathbb{R}^{S_{\eta}}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R}, \boldsymbol{\eta} \in \mathbb{R}_+^P \\ \text{subject to} \;\; & y_i(\langle \boldsymbol{w}_{\eta}, \Phi_{\eta}(\boldsymbol{x}_i)\rangle + b) \geq 1 - \xi_i \qquad \forall i. \end{aligned}$$

This problem, different from the primal problem of SimpleMKL, is not convex, but the solution strategy is the same. The objective function value of the primal formulation given $\boldsymbol{\eta}$ is used as the target function:

$$\begin{aligned} \text{minimize} \;\; & J(\boldsymbol{\eta}) = \frac{1}{2}\|\boldsymbol{w}_{\eta}\|_2^2 + C\sum_{i=1}^{N}\xi_i + r(\boldsymbol{\eta}) \\ \text{with respect to} \;\; & \boldsymbol{w}_{\eta} \in \mathbb{R}^{S_{\eta}}, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to} \;\; & y_i(\langle \boldsymbol{w}_{\eta}, \Phi_{\eta}(\boldsymbol{x}_i)\rangle + b) \geq 1 - \xi_i \qquad \forall i \end{aligned}$$

and the following dual formulation is used for the gradient step:

$$\begin{aligned}
\text{maximize} \quad & J(\boldsymbol{\eta}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) + r(\boldsymbol{\eta}) \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_{+}^{N} \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i.
\end{aligned}$$

The regularization function $r(\cdot)$ and $k_{\eta}(\cdot, \cdot)$ can be any differentiable function of $\boldsymbol{\eta}$ with continuous derivative. The gradient with respect to the kernel weights is calculated as

$$\frac{\partial J(\boldsymbol{\eta})}{\partial \eta_m} = \frac{\partial r(\boldsymbol{\eta})}{\partial \eta_m} - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \eta_m} \qquad \forall m.$$

Gender identification experiments on a face image data set are performed by combining kernels calculated on each individual feature, and hence, features used in kernels whose η_m goes to 0 are eliminated (Varma and Ray, 2007; Varma and Babu, 2009). The standard MKL and GMKL are trained with the kernel functions $k_{\eta}^{S}(\cdot, \cdot)$ and $k_{\eta}^{P}(\cdot, \cdot)$, respectively.

$$\begin{aligned}
k_{\eta}^{S}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{m=1}^{D} \eta_m \exp\left(-\gamma_m \left(\boldsymbol{x}_i[m] - \boldsymbol{x}_j[m]\right)^2\right) \\
k_{\eta}^{P}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \prod_{m=1}^{D} \exp\left(-\eta_m \left(\boldsymbol{x}_i[m] - \boldsymbol{x}_j[m]\right)^2\right) = \exp\left(\sum_{m=1}^{D} -\eta_m \left(\boldsymbol{x}_i[m] - \boldsymbol{x}_j[m]\right)^2\right)
\end{aligned}$$

where $[\cdot]$ indexes the elements of a vector. They show that GMKL with $k_{\eta}^{P}(\cdot, \cdot)$ performs significantly better than the standard MKL with $k_{\eta}^{S}(\cdot, \cdot)$. We see that using $k_{\eta}^{P}(\cdot, \cdot)$ as the combined kernel function is equivalent to using different scaling parameters on each feature and using an RBF kernel over these scaled features with unit radius, as done by Grandvalet and Canu (2003).

Cortes *et al.* (2010) develop a nonlinear kernel combination method based on kernel ridge regression and polynomial combination of kernels. They propose to combine kernels as follows:

$$k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\boldsymbol{q} \in \mathcal{Q}} \eta_{q_1 q_2 \ldots q_P} \prod_{m=1}^{P} k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)^{q_m}$$

where $\mathcal{Q} = \{\boldsymbol{q} : \boldsymbol{q} \in \mathbb{Z}_+^P, \;\; \sum_{m=1}^{P} q_m \leq d\}$ and $\eta_{q_1 \ldots q_P} \geq 0$. The number of parameters to be learned is too large and the combined kernel is simplified in order to reduce the learning complexity:

$$k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\boldsymbol{q} \in \mathcal{R}} \prod_{m=1}^{P} \eta_m^{q_m} k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)^{q_m}$$

where $\mathcal{R} = \{\boldsymbol{q} : \boldsymbol{q} \in \mathbb{Z}_+^P, \;\; \sum_{m=1}^{P} q_m = d\}$ and $\boldsymbol{\eta} \in \mathbb{R}^P$. The combination weights are optimized using the following min-max optimization problem:

$$\underset{\boldsymbol{\eta} \in \mathcal{M}}{\text{minimize}} \;\; \underset{\boldsymbol{\alpha} \in \mathbb{R}^N}{\text{maximize}} \;\; -\boldsymbol{\alpha}^\top (\mathbf{K}_{\eta} + \lambda \mathbf{I}) \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \boldsymbol{y}.$$

where $\mathcal{M}$ is a positive, bounded, and convex set. A projection-based gradient-descent algorithm can be utilized to solve this problem.

2.3. Combining Kernels Using Kernel Similarity Measures

We can learn the kernel combination weights by using a quality measure that gives performance estimates for the kernel matrices calculated on training data. This corresponds to a function that assigns weights to kernel functions:

$$\eta_m = g_{\eta}(\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_P).$$

2.3.1. Combining Kernels Using Kernel Alignment

Cristianini *et al.* (2002) define a notion of similarity between two kernels called *kernel alignment.* The empirical alignment of two kernels is calculated as follows:

$$\mathrm{A}(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle_F \langle \mathbf{K}_2, \mathbf{K}_2 \rangle_F}}$$

where $\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F = \sum_{i=1}^{N} \sum_{j=1}^{N} k_1(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1) k_2(\boldsymbol{x}_i^2, \boldsymbol{x}_j^2)$. This similarity measure can be seen as the cosine of angle between $\mathbf{K}_1$ and $\mathbf{K}_2$. $\boldsymbol{y}\boldsymbol{y}^\top$ can be defined as *ideal kernel* for binary classification, and the alignment between a kernel and the ideal kernel becomes

$$\mathrm{A}(\mathbf{K}, \boldsymbol{y}\boldsymbol{y}^\top) = \frac{\langle \mathbf{K}, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F}{\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F \langle \boldsymbol{y}\boldsymbol{y}^\top, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F}} = \frac{\langle \mathbf{K}, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F}{N\sqrt{\langle \mathbf{K}, \mathbf{K} \rangle_F}}.$$

Kernel alignment has one key property due to concentration (i.e., the probability of deviation from the mean is decayed exponentially), which enables us to keep the high alignment on a test set when we optimize it on a training set.

2.3.1.1. Linear Combination. Lanckriet *et al.* (2004a) propose to optimize the kernel alignment as follows:

$$\begin{aligned} \text{maximize} \;\; & \mathrm{A}(\mathbf{K}_\eta^{\text{tra}}, \boldsymbol{y}\boldsymbol{y}^\top) \\ \text{with respect to} \;\; & \mathbf{K}_\eta \in \mathcal{K}_L \\ \text{subject to} \;\; & \mathrm{tr}(\mathbf{K}_\eta) = 1 \end{aligned}$$

where the trace of the combined kernel is set to 1. This problem is equivalent to

$$\begin{aligned} \text{maximize} \;\; & \langle \mathbf{K}_\eta^{\text{tra}}, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F \\ \text{with respect to} \;\; & \mathbf{K}_\eta \in \mathcal{K}_L \\ \text{subject to} \;\; & \langle \mathbf{K}_\eta, \mathbf{K}_\eta \rangle_F = 1 \\ & \mathrm{tr}(\mathbf{K}_\eta) = 1 \end{aligned}$$

and this problem can be converted into the following SDP problem:

$$\begin{aligned} \text{maximize} \quad & \langle \mathbf{K}_{\eta}^{\text{tra}}, \boldsymbol{y}\boldsymbol{y}^{\top} \rangle_F \\ \text{with respect to} \quad & \mathbf{K}_{\eta} \in \mathcal{K}_L, \mathbf{A} \in \mathcal{S}^N \\ \text{subject to} \quad & \text{tr}(\mathbf{A}) \leq 1 \\ & \begin{pmatrix} \mathbf{A} & \mathbf{K}_{\eta}^{\top} \\ \mathbf{K}_{\eta} & \mathbf{I} \end{pmatrix} \succeq 0. \end{aligned}$$

Igel *et al.* (2007) propose maximizing the kernel alignment using gradient-based optimization. They calculate the gradient of the alignment with respect to the kernel parameters:

$$\frac{\partial A(\mathbf{K}_{\eta}, \boldsymbol{y}\boldsymbol{y}^{\top})}{\partial \eta_m} = \frac{\left\langle \dfrac{\partial \mathbf{K}_{\eta}}{\partial \eta_m}, \boldsymbol{y}\boldsymbol{y}^{\top} \right\rangle_F \langle \mathbf{K}_{\eta}, \mathbf{K}_{\eta} \rangle_F - \langle \mathbf{K}_{\eta}, \boldsymbol{y}\boldsymbol{y}^{\top} \rangle_F \left\langle \dfrac{\partial \mathbf{K}_{\eta}}{\partial \eta_m}, \mathbf{K}_{\eta} \right\rangle_F}{N\sqrt{\langle \mathbf{K}_{\eta}, \mathbf{K}_{\eta} \rangle_F^3}}.$$

In a transcription initiation site detection task for bacterial genes, they obtain better results by optimizing the kernel weights of the combined kernel function that is composed of six sequence kernels, using the gradient above.

2.3.1.2. Nonnegative Linear Combination. (Lanckriet *et al.*, 2004a) restrict the kernel weights to be nonnegative, their SDP formulation reduces to the following QCQP problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m^{\text{tra}}, \boldsymbol{y}\boldsymbol{y}^{\top} \rangle_F \\ \text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}_{+}^{P} \\ \text{subject to} \quad & \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F \leq 1. \end{aligned} \tag{2.6}$$

Kandola *et al.* (2002) propose to maximize the alignment between a nonnegative linear combination of kernels and the ideal kernel. The alignment can be calculated as follows:

$$\mathrm{A}(\mathbf{K}_{\eta}, \boldsymbol{y}\boldsymbol{y}^{\top}) = \frac{\sum\limits_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^{\top} \rangle_F}{N\sqrt{\sum\limits_{m=1}^{P} \sum\limits_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F}}.$$

We should choose kernel weights that maximize the alignment and this idea can be cast into the following optimization problem:

$$\begin{aligned} \text{maximize} \ \ & \mathrm{A}(\mathbf{K}_{\eta}, \boldsymbol{y}\boldsymbol{y}^{\top}) \\ \text{with respect to} \ \ & \boldsymbol{\eta} \in \mathbb{R}_{+}^{P} \end{aligned}$$

and this problem is equivalent to

$$\begin{aligned} \text{maximize} \ \ & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^{\top} \rangle_F \\ \text{with respect to} \ \ & \boldsymbol{\eta} \in \mathbb{R}_{+}^{P} \\ \text{subject to} \ \ & \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F = c. \end{aligned}$$

Using the Lagrangian function, we can convert it into the following unconstrained optimization problem:

$$\begin{aligned} \text{maximize} \ \ & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^{\top} \rangle_F - \mu \left(\sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - c \right) \\ \text{with respect to} \ \ & \boldsymbol{\eta} \in \mathbb{R}_{+}^{P}. \end{aligned}$$

Kandola *et al.* (2002) take $\mu = 1$ arbitrarily and add a regularization term to the objective function in order to prevent overfitting. The resulting QP is very similar to the hard margin SVM problem and is expected to give sparse kernel weights.

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F - \sum_{m=1}^{P}\sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - \lambda \sum_{m=1}^{P} \eta_m^2 \\ \text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}_+^P \end{aligned}$$

where we only learn the kernel combination weights.

<u>2.3.1.3. Convex Combination.</u> Qiu and Lane (2009) propose the following simple heuristic to select the kernel weights using kernel alignment:

$$\eta_m = \frac{\mathrm{A}(\mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^\top)}{\sum\limits_{h=1}^{P} \mathrm{A}(\mathbf{K}_h, \boldsymbol{y}\boldsymbol{y}^\top)} \qquad \forall m. \tag{2.7}$$

2.3.2. Combining Kernels Using Other Similarity Measures

He *et al.* (2008) choose to optimize the distance between the combined kernel matrix and the ideal kernel, instead of optimizing the kernel alignment measure, by using the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \langle \mathbf{K}_\eta - \boldsymbol{y}\boldsymbol{y}^\top, \mathbf{K}_\eta - \boldsymbol{y}\boldsymbol{y}^\top \rangle_F^2 \\ \text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1. \end{aligned}$$

This problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & \sum_{m=1}^{P} \sum_{h=1}^{P} \eta_m \eta_h \langle \mathbf{K}_m, \mathbf{K}_h \rangle_F - 2 \sum_{m=1}^{P} \eta_m \langle \mathbf{K}_m, \boldsymbol{y}\boldsymbol{y}^\top \rangle_F \\ \text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1. \end{aligned} \tag{2.8}$$

Nguyen and Ho (2008) propose another quality measure called feature space-based kernel matrix evaluation measure (FSM) defined as

$$\text{FSM}(\mathbf{K}, \boldsymbol{y}) = \frac{s_+ + s_-}{\|\boldsymbol{m}_+ - \boldsymbol{m}_-\|_2}$$

where $\{s_+, s_-\}$ are standard deviations of the positive and negative classes, and $\{\boldsymbol{m}_+, \boldsymbol{m}_-\}$ are class centers in the feature space. Tanabe *et al.* (2008) optimize the kernel weights for the convex combination of kernels by minimizing this measure:

$$\begin{aligned} \text{minimize} \quad & \text{FSM}(\mathbf{K}_\eta, \boldsymbol{y}) \\ \text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}_+^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1. \end{aligned}$$

This method give similar performance results when compared to the SMO-like algorithm of Bach *et al.* (2004) for a protein-protein interaction prediction problem using much less time and memory.

Ying *et al.* (2009) follow an information theoretic approach based on the Kullback-Leibler (KL) divergence between the combined kernel matrix and the optimal kernel matrix:

$$\underset{\boldsymbol{\eta}}{\text{minimize}} \quad \text{KL}(\mathcal{N}(\mathbf{0}, \mathbf{K}_\eta) \| \mathcal{N}(\mathbf{0}, \boldsymbol{y}\boldsymbol{y}^\top)).$$

The kernel combinations weights can be optimized by using a projected gradient-descent method.

2.4. Combining Kernels Using Boosting

Another possibility is to iteratively update by adding a new kernel to $f_\eta(\cdot)$ as training continues.

Inspired from ensemble (Kuncheva, 2004) and boosting methods (Schapire, 1990), Bennett *et al.* (2002) modify the decision function in order to use multiple kernels:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \sum_{m=1}^{P} \alpha_i^m k_m(\boldsymbol{x}_i^m, \boldsymbol{x}^m) + b.$$

The model parameters, $\{\boldsymbol{\alpha}^m\}_{m=1}^P$ and b, of kernel ridge regression model are learned using a gradient-descent algorithm in the function space. The columns of the combined kernel matrix are generated from the heterogeneous kernels on the fly. Bi *et al.* (2004) develop column generation boosting methods for binary classification and regression estimation problems. In each iteration, the proposed methods solve an LP or a QP on a working set depending on the regularization term used.

Crammer *et al.* (2003) modify the boosting methodology to work with kernels by rewriting two loss functions for a pair of data instances by considering the pair as a single instance:

$$\begin{aligned} \text{ExpLoss}(k(\boldsymbol{x}_i, \boldsymbol{x}_j), y_i y_j) &= \exp(-y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)) \\ \text{LogLoss}(k(\boldsymbol{x}_i, \boldsymbol{x}_j), y_i y_j) &= \log(1 + \exp(-y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j))). \end{aligned}$$

We iteratively update the combined kernel matrix by using one of these two loss functions.

2.5. Bayesian Kernel Combination

In a trained combiner parameterized by $\boldsymbol{\Theta}$, if we assume $\boldsymbol{\Theta}$ are random variables with a prior, we can use a Bayesian approach. For the case of a weighted sum, we can for example have a prior on the kernel weights, $\{\eta_m\}_{m=1}^P$.

Girolami and Rogers (2005) formulate a Bayesian hierarchical model and derive variational Bayes estimators for classification and regression problems. The proposed decision function is

$$f(\boldsymbol{x}) = \sum_{i=0}^{N} \alpha_i \sum_{m=1}^{P} \eta_m k_m(\boldsymbol{x}_i^m, \boldsymbol{x}^m)$$

where $\boldsymbol{\eta}$ is modeled with a Dirichlet prior and $\boldsymbol{\alpha}$ is modeled with a zero-mean Gaussian with an inverse gamma variance prior. Damoulas and Girolami (2009b) extend this method by adding auxiliary variables and developing a Gibbs sampler. Multinomial probit likelihood is used to obtain an efficient sampling procedure. These methods are applied to different bioinformatics problems, such as protein fold recognition and remote homology problems, and improve the prediction performances for these tasks (Damoulas and Girolami, 2008; Damoulas and Girolami, 2009a).

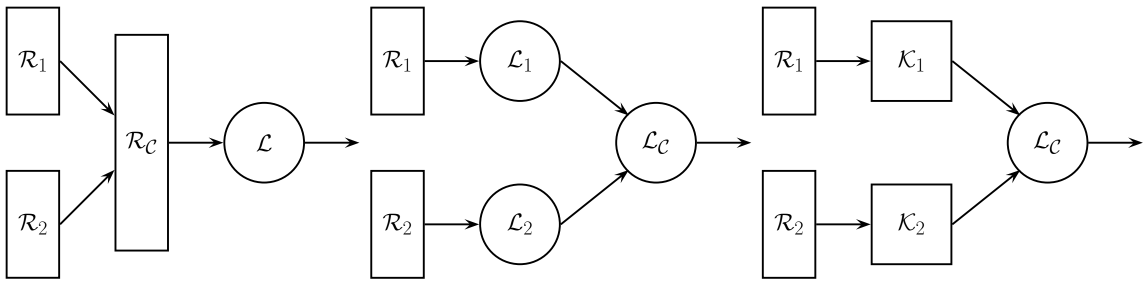
Girolami and Zhong (2007) use the kernel combination idea for the covariance matrices in Gaussian processes. Instead of using a single covariance matrix, they define a weighted sum of covariance matrices calculated over different data sources. A joint inference is performed for both the Gaussian process coefficients and the kernel combination weights.

2.6. Multiple Kernel Learning as Intermediate Integration

No single machine learning algorithm nor feature representation, in classification or regression, induces always the most accurate learner in any domain. The usual approach is to try many and choose the one that performs the best on a separate validation set unused during training. Recently, it has been shown that accuracy may be improved by combining multiple learners (Kuncheva, 2004). There are three possible methods for combining multiple feature representations: *early integration*, *late integration*, and *intermediate integration* (Noble, 2004).

Early integration is used when there are multiple feature representations and they are concatenated as one large vector and a single learner (classifier or regressor) is used. In late integration, multiple classifiers/regressors are trained over different feature representations and their decisions are combined by a trained learner. These two approaches can be applied with any classification/regression algorithm.

Kernel machines allow combination in a third way, using multiple kernels; this is called intermediate integration. A kernel function basically measures similarity between two data instances and a single-kernel machine can combine separate kernels for different feature representations, instead of combining data before training a single learner (as in early integration) or combining decisions from multiple learners (as in late integration). Figure 2.1 compares integration methods for multiple feature representations.



(a) Early integration (b) Late integration (c) Intermediate integration

Figure 2.1. Integration methods for multiple feature representations.

In general, we would expect early integration to suffer more from the curse of dimensionality when many input sources are concatenated. Late integration combines decisions and therefore is expected to be more robust; the disadvantage would be the need to train/store/use multiple learners. Intermediate integration is in between these two extremes where separate feature representations are not used in a raw manner (as in early integration) nor are decisions extracted from them (as in late integration) but are converted to similarities (using kernels) and fed to a single learner. An advantage of intermediate integration is in knowledge extraction; the relative importance of feature representations can be measured in terms of the weights assigned to the corresponding kernels.

2.7. Regularizing Multiple Kernel Learning Using Response Surface Methodology

The MKL formulation of Bach *et al.* (2004) introduces new regularization parameters that are directly related to the sparsity of the solution obtained. The easiest approach of setting them all equal does not always work well because it implies giving equal a priori weight to all kernels. The approach in Bach *et al.* (2005) is a heuristic to simply estimate the weights but it is clear that ideally, these weights should also be trained in a coupled manner with the kernel machines.

We propose using Response Surface Methodology (RSM) on validation error to optimize these parameters using data and to obtain more regularized solutions (Gönen and Alpaydın, 2010c). We show that optimizing the regularization parameters using an RSM-based approach leads to more sparse kernel ensembles where some kernels are given zero weight without diminishing accuracy. Not using a kernel in the ensemble may be either because: (a) the notion of similarity used in the kernel function is not appropriate, or, (b) the data source used to calculate the kernel does not carry useful information. If a kernel function is assigned zero weight in the combination rule, it means that this specific kernel function or the data source used in this kernel function does not carry discriminative information for the problem and hence can be pruned.

2.7.1. Regularized Multiple Kernel Learning

We use the mathematical model in (2.4) developed by Bach *et al.* (2004) as our base model, where the regularization term in the objective function consists of $\{\boldsymbol{w}_m\}_{m=1}^{P}$ and $\{d_m\}_{m=1}^{P}$. In general, $\{d_m\}_{m=1}^{P}$ are treated as scaling factors to balance the scale differences between kernel function outputs. It is a common procedure to normalize the kernel outputs before training:

$$\overline{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_i)k(\boldsymbol{x}_j, \boldsymbol{x}_j)}} \tag{2.9}$$

and the trace of each normalized kernel becomes N. All $\{d_m\}_{m=1}^P$ values are taken as 1 after normalization; this strategy is used in Sonnenburg *et al.* (2006b).

The solution obtained from the optimization problem gives us a linear combination of kernel functions that satisfies $\sum_{m=1}^P d_m^2 \eta_m = 1$ obtained from the Karush-Kuhn-Tucker optimality conditions (Bach *et al.*, 2004). This equation defines upper bounds for $\{\eta_m\}_{m=1}^P$ and we can control the feasible region for $\{\eta_m\}_{m=1}^P$ by changing $\{d_m\}_{m=1}^P$.

We consider $\{d_m\}_{m=1}^P$ as regularization parameters and from this perspective, the training process should also find better $\{d_m\}_{m=1}^P$ for better regularization, instead of simply selecting them all equal to 1. In order to show the effect of $\{d_m\}_{m=1}^P$ on regularization, we create four toy data sets, which are mixtures of Gaussians consisting of 1200 data instances with different number of components (see Table 2.2) and we test the effect of different $\{d_m\}_{m=1}^P$ using MKL on these data sets.

We combine three kernels (k_L, k_P, and k_G) on toy data sets by taking d_L as 1 and changing d_P and d_G on a grid in the log scale. We use the second-degree ($q = 2$) polynomial kernel and estimate s in the Gaussian kernel as follows:

$$s = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{x}_{nn(i)}\|_2 \tag{2.10}$$

where $nn(i)$ is the index of the nearest neighbor of $\boldsymbol{x}_i$.

Figure 2.2 shows the plot of the misclassification error on each data set for different values of d_P and d_G. The complexity of the discriminant increases as we increase the number of components, which increases the need for nonlinear kernels. We see that on Gauss2 and Gauss3 data sets, the best classification performance is obtained when

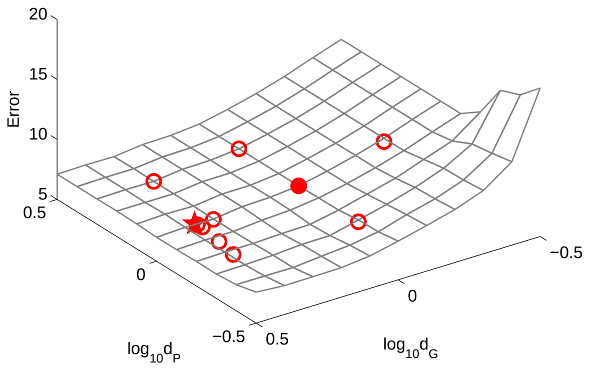
Table 2.2. Prior probabilities and Gaussian parameters used for toy data sets. In the GAUSS2 data set, each class has one Gaussian component. In the GAUSS3 data set, the positive class has two and the negative class has one component. The GAUSS4 data set has two components in each class and the GAUSS5 data set has three components in the positive class and two components in the negative class.

Data Set	Pos. Class (*priors*, *means*, *covariances*)	Neg. Class (*priors*, *means*, *covariances*)
GAUSS2	$p_1 = 0.50$ $\mu_1 = \begin{pmatrix} -1.0 \\ +1.0 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	$p_2 = 0.50$ $\mu_2 = \begin{pmatrix} +1.0 \\ -2.2 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$
GAUSS3	$p_1 = 0.25$ $\mu_1 = \begin{pmatrix} -2.0 \\ +1.0 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	$p_3 = 0.50$ $\mu_3 = \begin{pmatrix} +0.0 \\ -2.2 \end{pmatrix}$ $\Sigma_3 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$
	$p_2 = 0.25$ $\mu_2 = \begin{pmatrix} +2.0 \\ +1.0 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	
GAUSS4	$p_1 = 0.25$ $\mu_1 = \begin{pmatrix} -3.0 \\ +1.0 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	$p_3 = 0.25$ $\mu_3 = \begin{pmatrix} -1.0 \\ -2.2 \end{pmatrix}$ $\Sigma_3 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$
	$p_2 = 0.25$ $\mu_2 = \begin{pmatrix} +1.0 \\ +1.0 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	$p_4 = 0.25$ $\mu_4 = \begin{pmatrix} +3.0 \\ -2.2 \end{pmatrix}$ $\Sigma_4 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$
GAUSS5	$p_1 = 0.16$ $\mu_1 = \begin{pmatrix} -4.0 \\ +1.0 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	$p_4 = 0.25$ $\mu_4 = \begin{pmatrix} -2.0 \\ -2.2 \end{pmatrix}$ $\Sigma_4 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$
	$p_2 = 0.18$ $\mu_2 = \begin{pmatrix} +0.0 \\ +1.0 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	$p_5 = 0.25$ $\mu_5 = \begin{pmatrix} +2.0 \\ -2.2 \end{pmatrix}$ $\Sigma_5 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$
	$p_3 = 0.16$ $\mu_3 = \begin{pmatrix} +4.0 \\ +1.0 \end{pmatrix}$ $\Sigma_3 = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$	

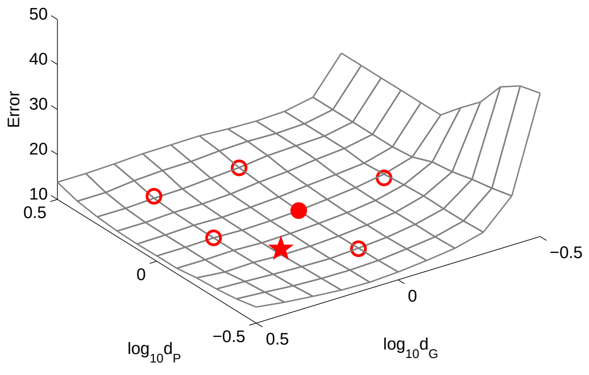
we choose d_P and d_G larger than or near 1, that is, when we force the model to use the more complex kernels k_P and k_G with smaller coefficients to avoid overfitting. On the other hand, d_P and d_G are chosen smaller than or near 1, resulting larger coefficients for k_P and k_G to avoid underfitting on GAUSS4 and GAUSS5 data sets. This indicates that each data set has its own smoothness constraint, and that $\{d_m\}_{m=1}^P$ should not just be taken equal but should be selected carefully to avoid overfitting or underfitting.

We propose to use RSM on validation error for selecting $\{d_m\}_{m=1}^P$ in an outer loop. RSM is a collection of statistical and mathematical techniques developed especially for process optimization (Myers and Montgomery, 2002). It is assumed that the system response, r, is written as some unknown function of P factors, $\boldsymbol{d} = \{d_m\}_{m=1}^P$:

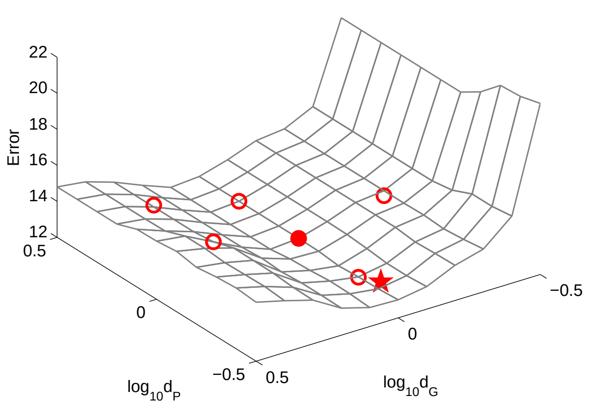
$$r = f(\{d_m\}_{m=1}^P) + \epsilon$$



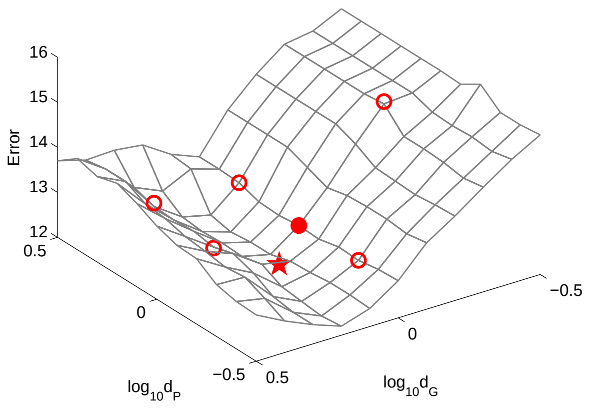
(a) Gauss2



(b) Gauss3



(c) Gauss4



(d) Gauss5

Figure 2.2. The average validation error over $\{d_m\}_{m=1}^P$ grid on toy data sets. We see that taking $\{d_m\}_{m=1}^P = 1$ (0 in the log scale) may not always be optimal (shown by a filled circle). The circles show the sampled points and the star shows the solution found by our proposed response surface approach.

where ϵ is a random error component. We do not know $f(\cdot)$ and instead, we approximate it by $\hat{f}(\cdot)$ using a low-order polynomial function (e.g., quadratic). We start by taking a small sample $(\mathbf{D}, \boldsymbol{r})$ of $\boldsymbol{d}$ and the corresponding r values, around some center in the $\boldsymbol{d}$ space. RSM consists of two basic steps: First, we fit the response surface $\hat{f}(\cdot)$ to $(\mathbf{D}, \boldsymbol{r})$ and then, we optimize the response, that is, from $\hat{f}(\cdot)$ we sample an $\boldsymbol{d}^*$ value, which we believe will return a better $f(\cdot)$ value. In the case of a quadratic fit, this can be calculated analytically as the optimum point of $\hat{f}(\cdot)$. $\boldsymbol{d}^*$ and the corresponding actual r^* are added to $(\mathbf{D}, \boldsymbol{r})$ and the procedure continues until there is no further improvement.

In our case, factors correspond to kernels and response corresponds to validation accuracy. We select a second-order model in the log scale (in order to ensure nonnegativity of $\{d_m\}_{m=1}^P$) for estimating the misclassification error. Without loss of generality, we can fix the regularization parameter for one of the kernels and the misclassification error can be expressed as

$$\beta_0 + \sum_{m=2}^{P} \beta_i \log_{10} d_m + \sum_{m=2}^{P} \sum_{h=m}^{P} \beta_{mh} \log_{10} d_m \log_{10} d_h$$

where β_0, $\{\beta_m\}_{m=1}^P$, and $\{\beta_{mh}\}_{m=1,h=m}^P$ are the model parameters.

At each iteration, because we are fitting a quadratic, at least $P(P+1)/2$ points are required to estimate the model parameters. To initialize, we use the second-order Koshal design (Myers and Montgomery, 2002) that uses three levels for each variable and with three factors (kernels), it is given as

$$\begin{array}{c} k_1 \\ k_2 \\ k_3 \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +\Delta & -\Delta & 0 & 0 & +\Delta \\ 0 & 0 & 0 & +\Delta & -\Delta & +\Delta \end{pmatrix}$$

where the rows correspond to kernels and the columns correspond to different $\{d_m\}_{m=1}^P$ values in the log scale assigned to kernels. For example, the first column is often referred as the "center point" in RSM (which in our case for the first iteration of RSM corresponds to having all $\{d_m\}_{m=1}^P$ equal to 1). Then, we construct "axial points" by moving towards negative and positive directions in each dimension by a small increment Δ (columns 2–5). The remaining runs required for estimating all model parameters are selected as "factorial points" by moving toward positive direction in each pair of dimensions (column 6); that is, we sample around the center point. We train MKLs at these points and check their misclassification error on the validation data. Then, we fit a quadratic to those set of errors and find analytically its optimum, which gives us the next sample point. We then sample there, add it to the list and make a new fit, until the improvement is negligible.

Using a second-order model in RSM corresponds to using Newton's method to find the minimum of the smoothed validation error curve. If the error curve obtained from sampled points is close to a quadratic surface, RSM converges very fast. If the error surface has an irregular shape with multiple local minima, RSM converges to one of the local minima like Newton's method.

The usual approach for finetuning parameters in machine learning is exhaustive grid search. There are two main motivations for using RSM instead of exhaustive grid search:

(a) RSM may require significantly fewer points: grid search requires $L^{(P-1)}$ points for the case with P kernels and L levels, whereas RSM starts with $P(P+1)/2$ initial points and converges long before $L^{(P-1)}$ iterations.
(b) RSM can obtain the result in terms of arbitrary factor values other than predefined factor levels and does a finer search than the resolution of the grid.

Figure 2.3 illustrates the idea of our proposed Regularized MKL (RMKL) in more detail. An initial search grid is constructed between Lines 2–11 using the second-order Koshal design. MKL is trained on validation sets with $\{d_m\}_{m=1}^{P}$ taken from the points of the initial search grid and validation errors are calculated for response surface calculations between Lines 12–16. The initial search grid points and their validation errors are used to start RSM and used to fit the quadratic response surface. The optimum operating point is calculated at its minimum. MKL method is trained at this point on the training set and its validation set (different from the training set) error is added to the sample points (Lines 17–27) and the next fit is done. RSM continues until convergence, which can be checked by $\|\boldsymbol{d}^{(t+1)} - \boldsymbol{d}^{(t)}\|_2 < \epsilon$ where ϵ is a small threshold.

An example is given in Figure 2.2. The stars show the points of convergence of RSM. We can see that it converges to points which are in deep valleys of the error function. The effect of regularization can also be clearly seen in Figure 2.4. MKL method selects a combination with weights $(\eta_L\text{-}\eta_P\text{-}\eta_G) = (0.00\text{-}0.25\text{-}0.75)$ on the

```
1: $gridSize \Leftarrow kernelCount(kernelCount + 1)/2$          start of initialization
2: $\mathbf{LD} \Leftarrow \mathbf{0}$                  center point ($\mathbf{0}$ vector in the log scale)
3: for $i = 2$ to $kernelCount$ do
4:     $\mathbf{LD} \Leftarrow [\mathbf{LD} \quad +\Delta \boldsymbol{e}_i]$          high level for each dimension
5:     $\mathbf{LD} \Leftarrow [\mathbf{LD} \quad -\Delta \boldsymbol{e}_i]$          low level for each dimension
6: end for
7: for $i = 2$ to $kernelCount$ do
8:     for $j = i + 1$ to $kernelCount$ do
9:         $\mathbf{LD} \Leftarrow [\mathbf{LD} \quad +\Delta \boldsymbol{e}_i + \Delta \boldsymbol{e}_j]$     high levels for each pair of dimensions
10:    end for
11: end for
12: for $t = 1$ to $gridSize$ do
13:     $\boldsymbol{d}^{(t)} \Leftarrow \begin{pmatrix} 1 \\ 10^{\mathbf{LD}(:,t)} \end{pmatrix}$   moves the point from the log scale to original space
14:     $ve^{(t)} \Leftarrow \text{TrainMKLSVM}(\boldsymbol{d}^{(t)})$   returns the average error on validation sets
15:     $\mathbf{VE} \Leftarrow [\mathbf{VE} \quad ve^{(t)}]$
16: end for                                                  end of initialization
17: loop
18:     $t \Leftarrow t + 1$
19:     $\boldsymbol{ld}^{(t)} \Leftarrow \text{ResponseSurface}(\mathbf{LD}, \mathbf{VE})$     fits surface and finds the best point
20:     $\boldsymbol{d}^{(t)} \Leftarrow \begin{pmatrix} 1 \\ 10^{\boldsymbol{ld}^{(t-1)}} \end{pmatrix}$   moves the point from the log scale to original space
21:     if $\|\boldsymbol{ld}^{(t)} - \boldsymbol{ld}^{(t-1)}\|_2 \leq \epsilon$ then                  checks convergence
22:         return $\boldsymbol{d}^{(t-1)}$                              returns the last point
23:     end if
24:     $ve^{(t)} \Leftarrow \text{TrainMKLSVM}(\boldsymbol{d}^{(t)})$   returns the average error on validation sets
25:     $\mathbf{LD} \Leftarrow [\mathbf{LD} \quad \boldsymbol{ld}^{(t)}]$
26:     $\mathbf{VE} \Leftarrow [\mathbf{VE} \quad \boldsymbol{ve}^{(t)}]$
27: end loop
```

Figure 2.3. Regularized Multiple Kernel Learning (RMKL)

Gauss3 data set and it overfits. When we train with our proposed RSM-based method, it converges to the point (d_L-d_P-d_G) = (1.00-0.85-1.46) selecting a different combination with weights (η_L-η_P-η_G) = (0.00-0.48-0.31), shown by a star in Figure 2.2(b). This

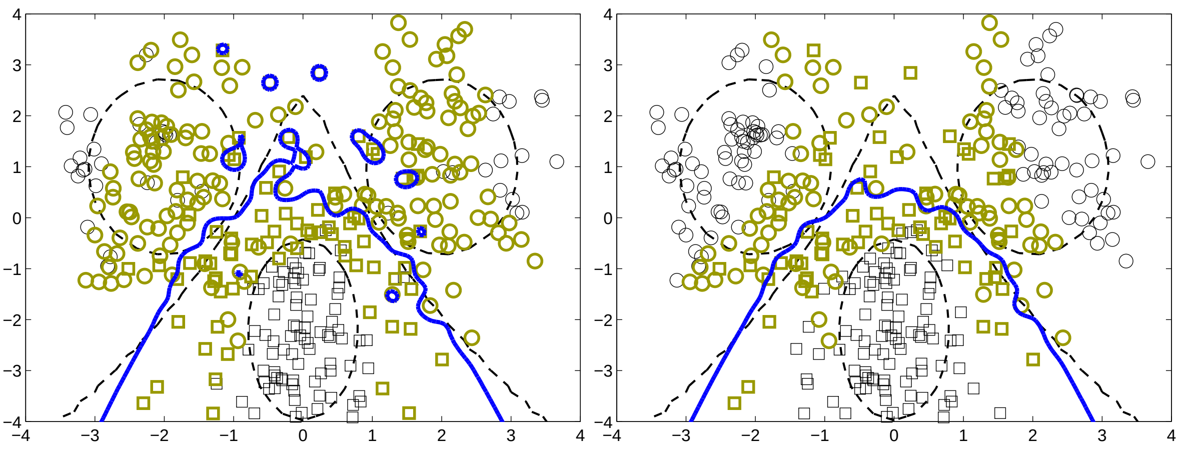


Figure 2.4. Separating hyperplanes (solid lines) and support vectors (thick points) of MKL and RMKL with (k_L-k_P-k_G) combination on the Gauss3 data set. Dashed lines show the Gaussians from which data are sampled and the optimal Bayes' discriminant. (a) (η_L-η_P-η_G) = (0.00-0.25-0.75) (b) (η_L-η_P-η_G) = (0.00-0.48-0.31), shown by a star in Figure 2.2(b).

particular run takes seven iterations where the first six iterations are used to initialize RSM. This solution implies that the second-order polynomial kernel is favored over the Gaussian kernel and this leads to a smaller combination weight for the Gaussian kernel and, as we see in Figure 2.4(a)-(b), a smoother separating boundary using fewer support vectors.

2.7.2. Discussion

Chapelle *et al.* (2002) propose a similar approach for choosing multiple parameters for SVMs. Their method tries to minimize the estimated test error bound and instead of explicitly fitting a response surface, updates parameters with a gradient-descent step calculated from the error bound. Our proposed method fits an approximate response surface for the test error using validation errors obtained over the sample points and finds the minimum point of the fitted response. Momma and Bennett (2002) use a more similar approach to select support vector regression parameters. They do not fit a response surface either; instead, they perform a moving grid search strategy by changing the center point of the grid.

RSM is also used by Blum *et al.* (2008) in protein structure prediction together with a Monte Carlo search procedure. Their model is not a kernel machine but they optimize the parameters of a specific function used in bioinformatics, called Rosetta energy function. They formulate the energy function in terms of input features and try to optimize it through optimizing the response surface. In order to get rid of irregularities emerging due to the high number of input features, they eliminate some of the features and calculate the response surface using the remaining features. In our case, the number of dimensions (factors) is limited by the number of kernel functions and this does not lead to any convergence problem in our experiments.

Regularization issues in multiple kernel learning have previously been studied. Bach *et al.* (2005) try to learn the entire regularization path for multiple kernel learning. The regularization path is calculated for the parameter that corresponds to the C parameter in the objective function of (2.4). We do not consider the optimization of C in the regularization process, we simply use a cross-validation procedure for this purpose, but it can also be added to this process by appending it as another dimension (factor) to RSM. The effect of $\{d_m\}_{m=1}^P$ onto regularization is also mentioned, though their optimization is not discussed. Micchelli and Pontil (2005) and Micchelli and Pontil (2007) formulate the multiple kernel learning problem from a different perspective; they directly perform optimization for convex combination parameters using square loss regularization.

Bach *et al.* (2005) and Bach (2008) also state that selecting the regularization parameters ($\{d_m\}_{m=1}^P$) in a data-dependent manner may lead to better results. For example, Bach *et al.* (2005) propose to select these parameters by looking at the eigenvalues of combined kernel matrices and their methodology is as follows: Given P different kernel matrices, the numbers ($\{e_m\}_{m=1}^P$) of the eigenvalues greater than 1/2 for each kernel matrix[1] are calculated and the regularization parameters are taken as $d_m = e_m^\gamma$ where γ is selected between 0 and 1 (we refer to this method as EMKL). In our experiments, we optimize γ by trying values $0, 0.1, 0.2, \ldots, 1$ on the validation sets

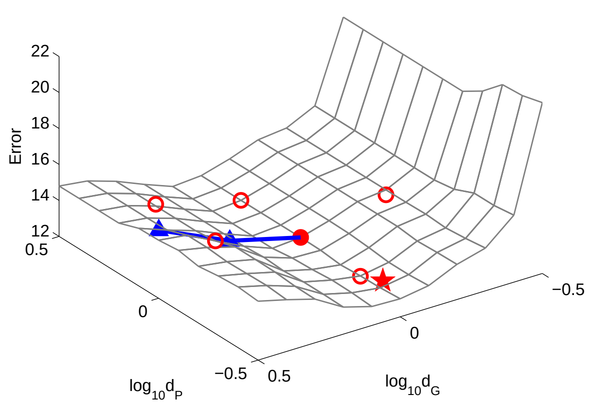
[1]In Bach *et al.* (2005), it was $1/2N$ but we normalize kernel matrices to unit diagonal instead of unit trace. So, we count the eigenvalues greater than 1/2.

of all folds and choosing the best. Our proposed method using RSM selects the regularization parameters by looking at the performance measures obtained with different parameter selections and does not consider any prior information.

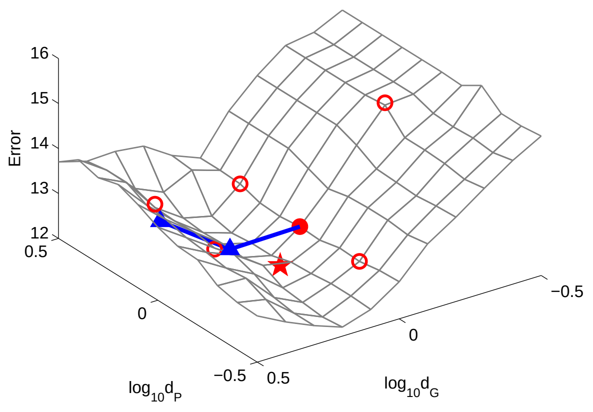
We also note that selecting the regularization parameters with the help of the eigenvalues integrates the kernel matrix complexity into the selection process before training. For example, the second-degree polynomial and the Gaussian kernel usually have higher $\{e_m\}_{m=1}^P$ values than the linear kernel and this leads to the penalization of these kernels if we choose γ larger than 0. Even if we use γ values smaller than 0, the regularization parameters are selected as a function of e_m values and this restricts us to use a predetermined region in the parameter space. Our method allows $\{d_m\}_{m=1}^P$ to converge to any point in the parameter space independently. The advantage of this difference can be clearly seen on Gauss4 and Gauss5 data sets in Figure 2.5. The search direction obtained by using the eigenvalues (shown by the line) may not be a good direction for searching the optimum point of the response surface (shown by the star). EMKL can improve the performance of MKL but it performs parameter selection only on this search direction and the selected parameter set may be suboptimal. RMKL selects $\{d_m\}_{m=1}^P$ from the whole parameter space by starting from a grid of samples (circles) around the center point (corresponding to the original MKL selection).

It is also possible to perform RSM on combination weights ($\{\eta_m\}_{m=1}^P$) directly by solving the canonical SVM optimization problems without using the MKL formulation; we refer to this method as RWKL. This approach clearly speeds up the training phase but we can not take advantage of the sparsity provided by the objective function of MKL formulation. The objective function of (2.4) forces the model to choose sparse kernel combinations, whereas replacing the kernel in the objective function of SVM with a weighted sum of kernels does not favor sparsity.

We are going to report our experimental results for comparing MKL, RMKL, EMKL, and RWKL in Subsection 5.2.2.



(a) Gauss4



(b) Gauss5

Figure 2.5. Misclassification errors over $\{d_m\}_{m=1}^P$ grid on Gauss4 and Gauss5 data sets. The circles and star show the sampled points and the solution found by our proposed RSM-based approach. The triangles and line show the sampled points and search direction if we use the eigenvalues of combined kernel matrices, as used in Bach *et al.* (2005).

2.8. Cost-Conscious Multiple Kernel Learning

We can also think $\{d_m\}_{m=1}^P$ in (2.4) as the cost coefficients for using $\{k_m(\cdot,\cdot)\}_{m=1}^P$ (Gönen and Alpaydın, 2010a). There are two possible cases:

(a) We can combine different kernel functions and $\{d_m\}_{m=1}^P$ may be considered as the costs of evaluating kernels. For example, evaluating the Gaussian kernel function is more costly than evaluating the linear kernel. Here, the kernel cost is generally expressed in terms of the required processor time.

(b) We can combine different representations or modalities and $\{d_m\}_{m=1}^P$ may be considered as the costs of extracting/sensing the corresponding representations/signals. Each data representation has its own data acquisition cost and kernel function evaluation time due to its different dimensionality. Kernel combination should favor the cheaper and smaller data representations if they are sufficient for accurate classification. If a particular data representation is not selected (i.e., its η is 0) after the training phase, we are not required to collect and prepare data for this representation in the testing phase. So, by assigning higher costs, we can eliminate some of the data representations and therefore decrease the total cost and time for test examples, unless the costly kernels/representations are absolutely necessary for accuracy. For example, in speech recognition where additional to the usual acoustic input, if we also use visual lip image as another source, we need to make sure that its contribution to accuracy is worth the cost of acquiring and processing the image. Or in biometrics where we have multiple modalities (face, fingerprint, iris, signature), we only want to include those whose costs can be justified in terms of additional accuracy.

From this perspective, setting all weights equal to 1 corresponds to assuming equal costs for all kernels, which, in general, is not true. We need a measure to estimate the total cost for the testing phase based on the number of support vectors and the kernel functions selected in training, which we define as

$$c = \underbrace{\sum_{i=1}^{N} \mathbf{1}(\alpha_i > 0)}_{\substack{\text{the number of} \\ \text{support vectors}}} \underbrace{\sum_{m=1}^{P} \mathbf{1}(\eta_m > 0) \frac{d_m}{\sum\limits_{h=1}^{P} d_h}}_{\substack{\text{the total normalized} \\ \text{cost for active kernels}}}$$

where we multiply the number of support vectors and the summation of the normalized costs for active kernels.

We are going to report our experimental results for comparing MKL and its cost-conscious variant in Subsection 5.2.3.

3. LOCALIZED MULTIPLE KERNEL LEARNING

In this chapter, we propose a nonlinear MKL method called Localized MKL (LMKL) that is composed of a kernel-based learning algorithm and a parametric gating model to assign local weights to kernel functions. These two components are trained in a coupled manner using a two-step alternating optimization algorithm. We derive the learning algorithm for three different gating models (softmax, sigmoid, and Gaussian) and apply the LMKL framework to four different machine learning problems (binary classification, regression, multiclass classification, and one-class classification).

Using a fixed (unweighted or weighted) sum assigns the same weight to a kernel over the whole input space. Assigning different weights to a kernel in different regions of the input space may produce a better classifier. If the data has underlying local structure, different similarity measures may be suited in different regions. We propose to divide the input space into regions using a gating function and assign combination weights to kernels in a data-dependent way (Gönen and Alpaydın, 2008); in the neural network literature, a similar architecture is previously proposed under the name "mixture of experts" (Jacobs *et al.*, 1991). The discriminant function for binary classification is rewritten as

$$f(\boldsymbol{x}) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}|\mathbf{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}^m) \rangle + b \qquad (3.1)$$

where $\eta_m(\boldsymbol{x}|\mathbf{V})$ is a parametric gating model that assigns a weight to $\Phi_m(\boldsymbol{x}^m)$ as a function of $\boldsymbol{x}$ and $\mathbf{V}$ is the matrix of gating model parameters. Note that unlike in MKL, in LMKL, it is not obligatory to combine different feature spaces; we can also use multiple copies of the same feature space (i.e., kernel) in different regions of the input space and thereby obtain a more complex discriminant function. For example, we can combine multiple linear kernels to get a piecewise linear discriminant.

3.1. Gating Models

We can use different gating models to assign kernel weights in a data-dependent way. The first gating model we investigate is the softmax gating model:

$$\eta_m(\boldsymbol{x}|\mathbf{V}) = \frac{\exp(\langle \boldsymbol{v}_m, \boldsymbol{x}^{\mathcal{G}} \rangle + v_{m0})}{\sum\limits_{h=1}^{P} \exp(\langle \boldsymbol{v}_h, \boldsymbol{x}^{\mathcal{G}} \rangle + v_{h0})} \qquad \forall m \tag{3.2}$$

where $\boldsymbol{x}^{\mathcal{G}} \in \mathbb{R}^{D_{\mathcal{G}}}$ is the representation of the input instance in the feature space in which we learn the gating model and $\mathbf{V} \in \mathbb{R}^{P \times (D_{\mathcal{G}}+1)}$ contains the gating model parameters $\{\boldsymbol{v}_m, v_{m0}\}_{m=1}^{P}$. The softmax gating model uses kernels in a competitive manner and generally a single kernel is active for each input. We may also use the sigmoid function instead of softmax and thereby allow multiple kernels to be used in a cooperative manner:

$$\eta_m(\boldsymbol{x}|\mathbf{V}) = \frac{1}{1 + \exp(-\langle \boldsymbol{v}_m, \boldsymbol{x}^{\mathcal{G}} \rangle - v_{m0})} \qquad \forall m. \tag{3.3}$$

Instead of parameterizing the boundaries of the local regions for kernels, we can also parameterize their centers and spreads using Gaussian gating:

$$\eta_m(\boldsymbol{x}|\mathbf{V}) = \frac{\exp(-\|\boldsymbol{x}^{\mathcal{G}} - \boldsymbol{\mu}_m\|_2^2/\sigma_m^2)}{\sum\limits_{h=1}^{P} \exp(-\|\boldsymbol{x}^{\mathcal{G}} - \boldsymbol{\mu}_h\|_2^2/\sigma_h^2)} \qquad \forall m \tag{3.4}$$

where $\mathbf{V} \in \mathbb{R}^{P \times (D_{\mathcal{G}}+1)}$ contains the means, $\{\boldsymbol{\mu}_m\}_{m=1}^{P}$, and the spreads, $\{\sigma_m\}_{m=1}^{P}$.

If we combine the same feature representation with different kernels (i.e., $\boldsymbol{x} = \boldsymbol{x}^1 = \boldsymbol{x}^2 = \ldots = \boldsymbol{x}^P$), we can simply use it also in the gating model (i.e., $\boldsymbol{x}^{\mathcal{G}} = \boldsymbol{x}$) (Gönen and Alpaydın, 2008). If we combine different feature representations with the same kernel, the gating model representation $\boldsymbol{x}^{\mathcal{G}}$ can be one of the representations, $\{\boldsymbol{x}^m\}_{m=1}^{P}$, a concatenation of a subset of them, or a completely different representation. In some application areas such as bioinformatics where data instances may appear in a non-vectorial format such as sequences, trees, and graphs, where we can calculate kernel matrices but can not represent the data instances as $\boldsymbol{x}$ vectors directly, we may use an empirical kernel map (Schölkopf *et al.*, 2004b) and define $\boldsymbol{x}^{\mathcal{G}}$ in terms of the kernel values (Gönen and Alpaydın, 2009c):

$$\boldsymbol{x}^{\mathcal{G}} = \begin{pmatrix} k_{\mathcal{G}}(\boldsymbol{x}_1, \boldsymbol{x}) & k_{\mathcal{G}}(\boldsymbol{x}_2, \boldsymbol{x}) & \cdots & k_{\mathcal{G}}(\boldsymbol{x}_N, \boldsymbol{x}) \end{pmatrix}^{\top} \tag{3.5}$$

where the gating kernel, $k_{\mathcal{G}}(\cdot, \cdot)$, can be one of the combined kernels, $\{k_m(\cdot, \cdot)\}_{m=1}^{P}$, a combination of them, or a completely different kernel used only for determining the gating boundaries.

3.2. Mathematical Model

By using the discriminant function in (3.1) and regularizing the discriminant coefficients of all the feature spaces together, LMKL obtains the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \sum_{m=1}^{P} \|\boldsymbol{w}_m\|_2^2 + C \sum_{i=1}^{N} \xi_i \\ \text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{S_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, \mathbf{V} \in \mathbb{R}^{P \times (D_{\mathcal{G}}+1)}, b \in \mathbb{R} \\ \text{subject to} \quad & y_i \left(\sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i | \mathbf{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i^m) \rangle + b \right) \geq 1 - \xi_i \qquad \forall i \end{aligned} \tag{3.6}$$

where nonconvexity is introduced to the model due to the nonlinearity formed by using the gating model outputs in the separation constraints. Instead of trying to solve (3.6) directly, we can use a two-step alternating optimization algorithm (Gönen and Alpaydın, 2008), also used for choosing kernel parameters (Chapelle *et al.*, 2002) and obtaining η_m parameters of MKL (Rakotomamonjy *et al.*, 2008). This procedure consists of two basic steps: (a) solving the model with a fixed gating model, and, (b) updating the gating model parameters with the gradients calculated from the current solution.

For a fixed $\mathbf{V}$, we obtain the Lagrangian dual of the primal problem (3.6) as follows:

$$\begin{aligned} L_D(\mathbf{V}) = \frac{1}{2}\sum_{m=1}^{P} \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\beta_i\xi_i \\ - \sum_{i=1}^{N}\alpha_i\left(y_i\left(\sum_{m=1}^{P}\eta_m(\boldsymbol{x}_i|\mathbf{V})\langle\boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i^m)\rangle + b\right) - 1 + \xi_i\right) \end{aligned}$$

and taking the derivatives of $L_D(\mathbf{V})$ with respect to the primal variables gives

$$\begin{aligned} \frac{\partial L_D(\mathbf{V})}{\partial \boldsymbol{w}_m} &= 0 \Rightarrow \boldsymbol{w}_m = \sum_{i=1}^{N}\alpha_i y_i \eta_m(\boldsymbol{x}_i|\mathbf{V})\Phi_m(\boldsymbol{x}_i^m) \qquad \forall m \\ \frac{\partial L_D(\mathbf{V})}{\partial b} &= 0 \Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \frac{\partial L_D(\mathbf{V})}{\partial \xi_i} &= 0 \Rightarrow C = \alpha_i + \beta_i \qquad \forall i. \end{aligned} \tag{3.7}$$

From $L_D(\mathbf{V})$ and (3.7), the dual formulation is obtained as

$$\begin{aligned} \text{maximize} \quad & J(\mathbf{V}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_i y_i y_j k\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \qquad \forall i \end{aligned} \tag{3.8}$$

where the *locally combined kernel function* is defined as

$$k\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P}\eta_m(\boldsymbol{x}_i|\mathbf{V})\underbrace{\langle\Phi_m(\boldsymbol{x}_i^m), \Phi_m(\boldsymbol{x}_j^m)\rangle}_{k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)}\eta_m(\boldsymbol{x}_j|\mathbf{V}). \tag{3.9}$$

By using the support vector coefficients obtained from (3.8) and the gating model parameters, we obtain the following discriminant function:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i k_\eta(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

Due to strong duality, for a given $\mathbf{V}$, the gradients of the objective function in (3.8) are equal to the gradients of the objective function in (3.6). These gradients are used to update the gating model parameters at each step.

3.3. Training with Alternating Optimization

We can find the gradients of $J(\mathbf{V})$ with respect to the parameters of all three gating models. The gradients of (3.8) with respect to the parameters of the softmax gating model (3.2) are

$$\begin{aligned}
\frac{\partial J(\mathbf{V})}{\partial \boldsymbol{v}_m} &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{h=1}^{P} \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V}) k_h(\boldsymbol{x}_i^h, \boldsymbol{x}_j^h) \eta_h(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad (\boldsymbol{x}_i^{\mathcal{G}}(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + \boldsymbol{x}_j^{\mathcal{G}}(\delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V}))) \\
\frac{\partial J(\mathbf{V})}{\partial v_{m0}} &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{h=1}^{P} \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V}) k_h(\boldsymbol{x}_i^h, \boldsymbol{x}_j^h) \eta_h(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad (\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V}) + \delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V}))
\end{aligned}$$

where $\Upsilon_i^j = \alpha_i \alpha_j y_i y_j$. The same gradients with respect to the parameters of the sigmoid gating model (3.3) are

$$\begin{aligned}
\frac{\partial J(\mathbf{V})}{\partial \boldsymbol{v}_m} &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \Upsilon_i^j \eta_m(\boldsymbol{x}_i|\mathbf{V}) k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) \eta_m(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad (\boldsymbol{x}_i^{\mathcal{G}}(1 - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + \boldsymbol{x}_j^{\mathcal{G}}(1 - \eta_m(\boldsymbol{x}_j|\mathbf{V}))) \\
\frac{\partial J(\mathbf{V})}{\partial v_{m0}} &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \Upsilon_i^j \eta_m(\boldsymbol{x}_i|\mathbf{V}) k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) \eta_m(\boldsymbol{x}_j|\mathbf{V}) \\
&\quad (1 - \eta_m(\boldsymbol{x}_i|\mathbf{V}) + 1 - \eta_m(\boldsymbol{x}_j|\mathbf{V}))
\end{aligned}$$

where the gating model parameters for a kernel function are updated independently. We can also find the gradients with respect to the means and the spreads of the Gaussian gating model (3.4) are

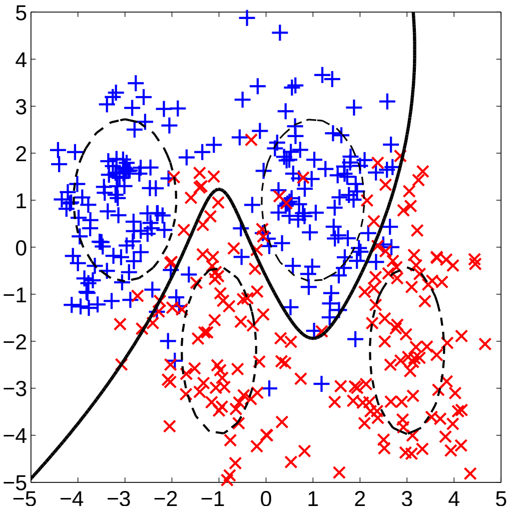
$$\begin{aligned}\frac{\partial J(\mathbf{V})}{\partial \boldsymbol{\mu}_m} &= -\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{h=1}^{P} \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V}) k_h(\boldsymbol{x}_i^h, \boldsymbol{x}_j^h) \eta_h(\boldsymbol{x}_j|\mathbf{V}) \\ &\quad ((\boldsymbol{x}_i^{\mathcal{G}} - \boldsymbol{\mu}_m)(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + (\boldsymbol{x}_j^{\mathcal{G}} - \boldsymbol{\mu}_m)(\delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V})))/\sigma_m^2 \\ \frac{\partial J(\mathbf{V})}{\partial \sigma_m} &= -\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{h=1}^{P} \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V}) k_h(\boldsymbol{x}_i^h, \boldsymbol{x}_j^h) \eta_h(\boldsymbol{x}_j|\mathbf{V}) \\ &\quad (\|\boldsymbol{x}_i^{\mathcal{G}} - \boldsymbol{\mu}_m\|_2^2(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + \|\boldsymbol{x}_j^{\mathcal{G}} - \boldsymbol{\mu}_m\|_2^2(\delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V})))/\sigma_m^3.\end{aligned}$$

The complete algorithm of our proposed LMKL is summarized in Figure 3.1. The convergence of the algorithm can be determined by observing the change in the objective function value of (3.8).

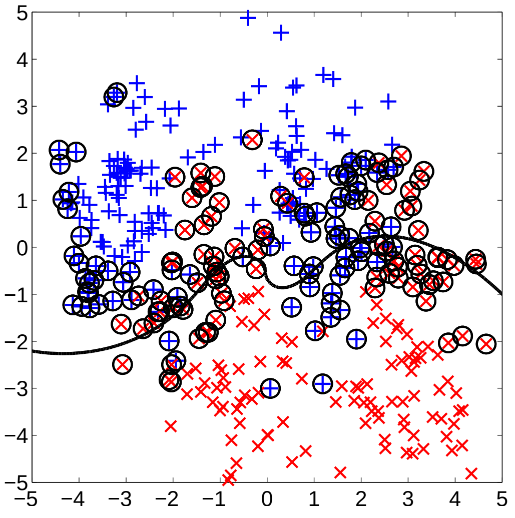
```
1: Initialize $\mathbf{V}^{(0)}$ randomly
2: repeat
3:    Calculate $\mathbf{K}_\eta^{(t)} = \{k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i,j=1}^N$ using $\mathbf{V}^{(t)}$
4:    Solve kernel machine with $\mathbf{K}_\eta^{(t)}$
5:    Calculate descent direction $\frac{\partial J(\mathbf{V})}{\partial \mathbf{V}}$
6:    Determine step size, $\Delta^{(t)}$, using Armijo's rule
7:    Update gating model parameters: $\mathbf{V}^{(t+1)} \Leftarrow \mathbf{V}^{(t)} - \Delta^{(t)}\frac{\partial J(\mathbf{V})}{\partial \mathbf{V}}$
8: until convergence
```

Figure 3.1. Localized Multiple Kernel Learning (LMKL)

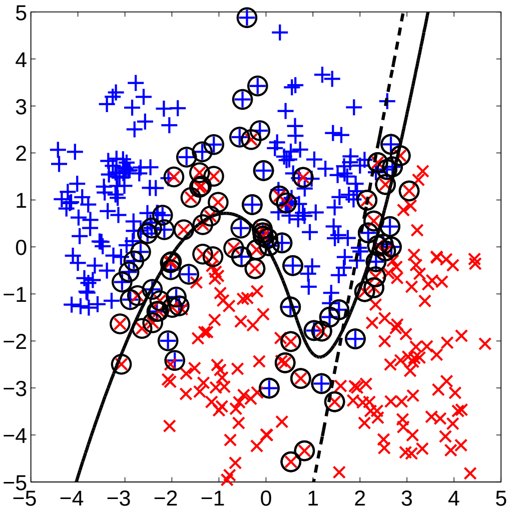
In order to illustrate our proposed algorithm, we use the toy data set Gauss4. First, we train both MKL and LMKL with softmax gating to combine a linear kernel, k_L, and a second-degree polynomial kernel, k_P ($q = 2$). Figure 3.2(b) shows the classification boundaries calculated and the support vectors stored on one of the training folds by MKL that assigns combination weights 0.32 and 0.68 to k_L and k_P, respectively. We see that using the kernel matrix obtained by combining k_L and k_P with these



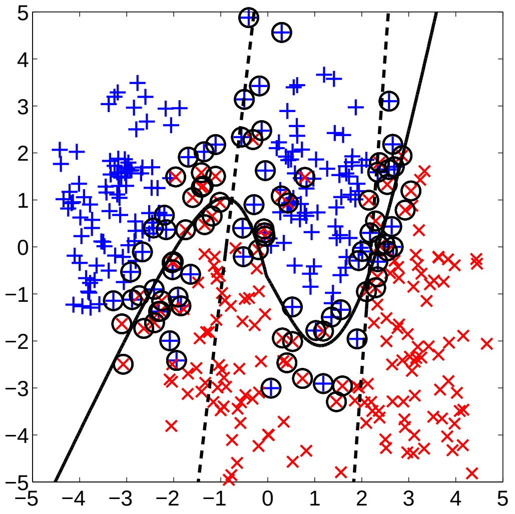
(a) Gauss4 Data Set



(b) MKL with (k_L-k_P)



(c) LMKL with (k_L-k_P)



(d) LMKL with (k_L-k_L-k_L)

Figure 3.2. MKL and LMKL solutions on the Gauss4 data set. (a) The dashed ellipses show the Gaussians from which data are sampled and the solid line shows the optimal Bayes' discriminant. (b)-(d) The solid lines show the discriminants learned. The circled data points represent the support vectors stored.

weights, we could not achieve a good approximation to the optimal Bayes' boundary. As we see in Figure 3.2(c), LMKL divides the input space into two regions (the gating boundary is shown as a dashed line) and uses the polynomial kernel to separate one component from two others quadratically in one region and the linear kernel for the other component in the other region. We see that we get a very good approximation of the optimal Bayes' boundary. The softmax function in the gating model achieves a smooth transition between the two kernels. The superiority of the localized approach is also apparent in the smoothness of the fit that uses fewer support vectors: MKL achieves 90.95±0.61 per cent average test accuracy by storing 38.23±2.34 per cent of training instances as support vectors, whereas LMKL achieves 91.83±0.24 per cent average test accuracy by storing 25.13±0.91 per cent support vectors.

With LMKL, we can also combine multiple copies of the same kernel, as shown in Figure 3.2(d), which shows the classification and gating model boundaries of LMKL (solid and dashed lines, respectively) using three linear kernels and approximates the optimal Bayes' boundary in a piecewise linear manner. For this configuration, LMKL achieves 91.78±0.55 per cent average test accuracy by storing 23.83±1.20 per cent support vectors. Instead of using complex kernels such as polynomial kernels of high-degree or the Gaussian kernel, local combination of simple kernels (e.g., linear or low-degree polynomial kernels) can produce accurate classifiers and avoid overfitting. Figure 3.3 shows the average test accuracies and support vector percentages with one standard deviation for LMKL with different number of linear kernels. We see that even if we provide more kernels than needed, LMKL uses only as many support vectors as required and does not overfit. LMKL obtains nearly the same average test accuracies and support vector percentages with three or more linear kernels.

3.4. Extensions to Other Algorithms

We extend our LMKL framework for binary classification to other kernel-based algorithms, namely SVR, MCSVM, and OCSVM. Note that any kernel machine that has a hyperplane-based decision function can be localized by replacing $\langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle$ with $\sum_{m=1}^{P} \eta_m(\boldsymbol{x}|\mathbf{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}^m) \rangle$ and deriving the corresponding update rules.

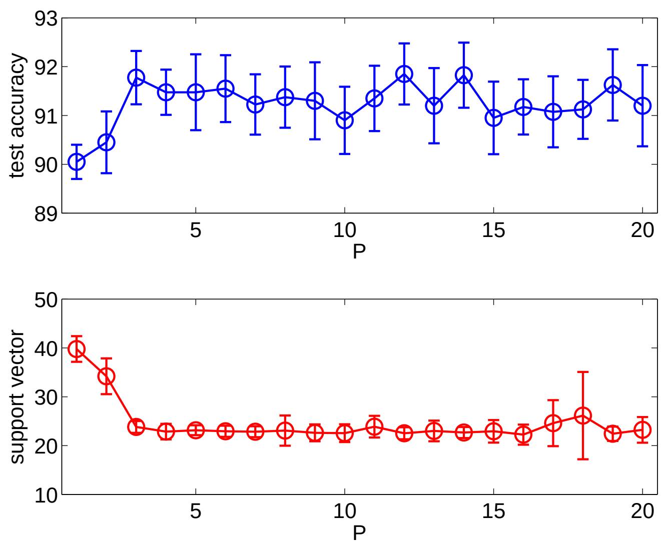


Figure 3.3. The average test accuracies and support vector percentages on the GAUSS4 data set obtained by LMKL with multiple copies of linear kernels and softmax gating.

3.4.1. Regression Support Vector Machines

We can also apply the localized kernel idea to ϵ-tube SVR (Gönen and Alpaydın, 2010b). The decision function is rewritten as

$$f(\boldsymbol{x}) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}|\mathbf{V})\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}^m)\rangle + b$$

and the modified primal optimization problem is

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{m=1}^{P} \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N}(\xi_i^+ + \xi_i^-) \\
\text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{S_m}, \boldsymbol{\xi}^+ \in \mathbb{R}_+^N, \boldsymbol{\xi}^- \in \mathbb{R}_+^N, \mathbf{V} \in \mathbb{R}^{P\times(D_{\mathcal{G}}+1)}, b \in \mathbb{R} \\
\text{subject to} \quad & \epsilon + \xi_i^+ \geq y_i - \sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i|\mathbf{V})\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i^m)\rangle - b \qquad \forall i \\
& \epsilon + \xi_i^- \geq \sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i|\mathbf{V})\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i^m)\rangle + b - y_i \qquad \forall i
\end{aligned}$$

where $\{\boldsymbol{\xi}^+, \boldsymbol{\xi}^-\}$ are the vectors of slack variables and ϵ is the width of the regression tube. For a given $\mathbf{V}$, the corresponding dual formulation is

$$
\begin{aligned}
\text{maximize} \quad & J(\mathbf{V}) = \sum_{i=1}^{N} y_i(\alpha_i^+ - \alpha_i^-) - \epsilon \sum_{i=1}^{N} (\alpha_i^+ + \alpha_i^-) \\
& \qquad - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
\text{with respect to} \quad & \boldsymbol{\alpha}^+ \in \mathbb{R}_+^N, \boldsymbol{\alpha}^- \in \mathbb{R}_+^N \\
\text{subject to} \quad & \sum_{i=1}^{N} (\alpha_i^+ - \alpha_i^-) = 0 \\
& C \geq \alpha_i^+ \geq 0 \qquad \forall i \\
& C \geq \alpha_i^- \geq 0 \qquad \forall i
\end{aligned}
$$

and the resulting decision function is

$$
f(\boldsymbol{x}) = \sum_{i=1}^{N} (\alpha_i^+ - \alpha_i^-) k_\eta(\boldsymbol{x}_i, \boldsymbol{x}) + b.
$$

The same learning algorithm given for binary classification problems can be applied to regression problems by simply replacing Υ_i^j in gradient-descent of the gating model (see Section 3.3) with $(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)$.

We illustrate the applicability of LMKL to regression problems on the MOTORCYCLE data set discussed by Silverman (1985). We train LMKL with three linear kernels and softmax gating ($C = 1000$ and $\epsilon = 16$) using 10-fold cross validation. Figure 3.4 shows the average of global and local fits obtained for these 10 folds. We learn a piecewise linear fit through three local models that are obtained using linear kernels in each region and we combine them by using the softmax gating model (shown by dashed lines). The softmax gating model divides the input space between kernels, generally selects a single kernel to use, and also ensures a smooth transition between local fits.

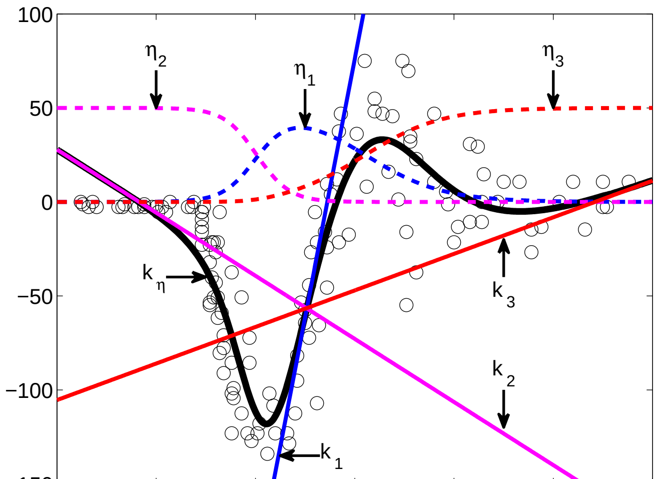


Figure 3.4. Global and local fits (solid lines) obtained by LMKL with three linear kernels and softmax gating on the MOTORCYCLE data set. The dashed lines show gating model outputs, which are multiplied by 50 for visual clarity.

3.4.2. One-Class Support Vector Machines

OCSVM is a discriminative method proposed for novelty detection problems (Schölkopf and Smola, 2002). The task is to learn the smoothest hyperplane that puts most of the training instances to one side of the hyperplane while allowing other instances remaining on the other side with a cost. In the localized version, we rewrite the discriminant function as

$$f(\boldsymbol{x}) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}|\mathbf{V})\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}^m)\rangle + b$$

and the modified primal optimization problem is

$$\begin{aligned} \text{minimize} \ & \frac{1}{2}\sum_{m=1}^{P} \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N} \xi_i + b \\ \text{with respect to} \ & \boldsymbol{w}_m \in \mathbb{R}^{S_m}, \boldsymbol{\xi} \in \mathbb{R}_+^N, \mathbf{V} \in \mathbb{R}^{P \times (D_{\mathcal{G}}+1)}, b \in \mathbb{R} \\ \text{subject to} \ & \sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i|\mathbf{V})\langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i^m)\rangle + b + \xi_i \geq 0 \qquad \forall i. \end{aligned}$$

For a given $\mathbf{V}$, we obtain the following dual optimization problem:

$$\begin{aligned} \text{maximize} \quad & J(\mathbf{V}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_i k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to} \quad & \sum_{i=1}^{N} \alpha_i = 1 \\ & C \geq \alpha_i \geq 0 \qquad \forall i \end{aligned}$$

and the resulting discriminant function is

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k_\eta(\boldsymbol{x}_i, \boldsymbol{x}) + b.$$

In the learning algorithm, Υ_i^j should be replaced with $\alpha_i \alpha_j$ when calculating the gradients with respect to the gating model parameters.

3.4.3. Multiclass Support Vector Machines

We can easily apply LMKL to the multimachine approach by solving (3.8) for each two-class problem separately. In such a case, we obtain different gating models parameters and hence, different kernel weighing strategies for each of the problems. Another possibility is to solve these uncoupled problems separately but learn a common gating model; a similar approach is used for obtaining common kernel weights in MKL for multiclass problems (Rakotomamonjy *et al.*, 2008).

For the single-machine approach, for class l, we write the discriminant function as follows:

$$f^l(\boldsymbol{x}) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}|\mathbf{V}) \langle \boldsymbol{w}_m^l, \Phi_m(\boldsymbol{x}^m) \rangle + b^l.$$

The modified primal optimization problem is

$$\begin{aligned}
\text{minimize} \;\; & \frac{1}{2}\sum_{m=1}^{P}\sum_{l=1}^{K}\|\boldsymbol{w}_m^l\|_2^2 + C\sum_{i=1}^{N}\sum_{l=1}^{K}\xi_i^l \\
\text{with respect to} \;\; & \boldsymbol{w}_m^l \in \mathbb{R}^{S_m}, \boldsymbol{\xi}^l \in \mathbb{R}_+^N, \mathbf{V} \in \mathbb{R}^{P\times(D_{\mathcal{G}}+1)}, b^l \in \mathbb{R} \\
\text{subject to} \;\; & f^{y_i}(\boldsymbol{x}_i) - f^l(\boldsymbol{x}_i) \geq 2 - \xi_i^l \qquad \forall(i, l \neq y_i) \\
& \xi_i^{y_i} = 0 \qquad \forall i.
\end{aligned}$$

We can obtain the dual formulation for a given $\mathbf{V}$ by following the same derivation steps:

$$\begin{aligned}
\text{maximize} \;\; & J(\mathbf{V}) = 2\sum_{i=1}^{N}\sum_{l=1}^{K}\alpha_i^l - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\delta_{y_i}^{y_j}A_iA_j - \sum_{l=1}^{K}\alpha_i^l(2\alpha_j^{y_i} - \alpha_j^l)\right)k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
\text{with respect to} \;\; & \boldsymbol{\alpha}^l \in \mathbb{R}_+^N \\
\text{subject to} \;\; & \sum_{i=1}^{N}\alpha_i^l - \sum_{i=1}^{N}\delta_{y_i}^l A_i = 0 \qquad \forall l \\
& (1-\delta_{y_i}^l)C \geq \alpha_i^l \geq 0 \qquad \forall(i,l)
\end{aligned}$$

where $A_i = \sum_{l=1}^{K}\alpha_i^l$. The resulting discriminant functions that use the locally combined kernel function are given as

$$f^l(\boldsymbol{x}) = \sum_{i=1}^{N}(\delta_{y_i}^l A_i - \alpha_i^l)k_\eta(\boldsymbol{x}_i, \boldsymbol{x}) + b^l.$$

Υ_i^j should be replaced with $(\delta_{y_i}^{y_j}A_iA_j - \sum_{l=1}^{K}\alpha_i^l(2\alpha_j^{y_i} - \alpha_j^l))$ in learning the gating model parameters for multiclass classification problems.

3.5. Discussion

In this section, we discuss the key properties of our proposed method and compare it with similar MKL methods in the literature.

3.5.1. Computational Complexity

When we are training LMKL, we need to solve a canonical kernel machine problem with the combined kernel obtained with the current gating model parameters and calculate the gradients of $J(\mathbf{V})$ at each iteration. The gradient calculation step has lower time complexity compared to the kernel machine solver. The gradients calculations are made by using the support vectors of the current iteration. The computational complexity of LMKL mainly depends on the complexity of the canonical kernel machine solver used in the main loop, which can be reduced by using a hot-start procedure (i.e., starting from the previous solution). The number of iterations before convergence clearly depends on the training data and the step size selection procedure. The key issue for faster convergence is to select good gradient-descent step sizes at each iteration. The step size of each iteration should be determined with a line search method (e.g., Armijo's rule), which requires solving additional kernel machine problems. Clearly, the time complexity for each iteration increases but the algorithm converges in fewer iterations. In practice, we see convergence in five to 20 iterations.

One main advantage of LMKL is in reducing the time complexity for the testing phase as a result of localization. When calculating the locally combined kernel function, $k_\eta(\boldsymbol{x}_i, \boldsymbol{x})$, in (3.9), $k_m(\boldsymbol{x}_i^m, \boldsymbol{x}^m)$ needs to be evaluated or calculated only if both $\eta_m(\boldsymbol{x}_i)$ and $\eta_m(\boldsymbol{x})$ are active, i.e., nonzero.

3.5.2. Knowledge Extraction

The kernel weights obtained by MKL can be used to extract knowledge about the relative contributions of kernel functions used in combination. Different kernels define different similarity measures and we can deduce which similarity measures are appropriate for the task at hand. If kernel functions are evaluated over different feature subsets or feature representations, the important ones have higher combination weights. With our LMKL framework, we can extract similar information for different regions of the input space. This enables us to extract information about kernels (similarity measures), feature subsets, and/or feature representations in a data-dependent manner.

3.5.3. Regularization

Canonical kernel machines learn sparse models as a result of regularization on the weight vector but the underlying complexity of the kernel function is the main factor for determining the model complexity. The main advantage of LMKL in terms of regularization over canonical kernel machines is the inherent regularization effect on the gating model in (3.7). When we regularize hyperplane weight vectors, we also regularize the gating model as a side effect. MKL can combine only different kernel functions and more complex kernels are favored over the simpler ones in order to get better performance. However, LMKL can also combine multiple copies of the same kernel and it can dynamically construct a more complex locally combined kernel by using the kernels in a data-dependent way. LMKL eliminates some of the kernels by assigning zero weights to the corresponding gating outputs in order to get a more regularized solution. Figure 3.3 gives empirical support to this regularization effect, where we see that LMKL does not overfit even if we increase the number of kernels up to 20.

3.5.4. Dimensionality Reduction

The localized kernel idea can also be combined with dimensionality reduction. If the training instances have a local structure (i.e., lie on low-dimensional manifolds locally), we can learn low-dimensional local projections in each region, which we can also use for visualization. Previously, it has been proposed to integrate a projection matrix into the discriminant function (Chapelle *et al.*, 2002) and we extend this idea to project data instances into different feature spaces by using local projection matrices combined with a gating model, and calculate the combined kernel function with the dot product in the combined feature space (see Chapter 4). The local projection matrices can be learned with the other parameters, as before, using a two-step alternating optimization algorithm.

3.5.5. Related Work

LMKL finds a nonlinear combination of kernel functions with the help of the gating model. The idea of learning a nonlinear combination is also discussed in different studies. For example, Lewis *et al.* (2006a) propose a latent variable generative model using the maximum entropy discrimination to learn data-dependent kernel combination weights. This method combines a generative probabilistic model with a discriminative large margin method using a log-ratio of Gaussian mixtures as the classifier.

In a more recent work, a nonlinear kernel combination method based on kernel ridge regression and polynomial combination of kernels is proposed and the kernel weights are optimized over a positive, bounded, and convex set using a projection-based gradient-descent algorithm (Cortes *et al.*, 2010).

Similar to LMKL, a Bayesian approach is developed for combining different feature representations in a data-dependent way under the Gaussian process framework (Christoudias *et al.*, 2009). A common covariance function is obtained by combining the covariances of feature representations in a nonlinear manner. This formulation can identify the noisy data instances for each feature representation and prevent them from being used. Classification is performed using the standard Gaussian processes approach with the common covariance function.

Inspired from LMKL, two methods that learn a data-dependent kernel function are used for image recognition applications (Yang *et al.*, 2009; Yang *et al.*, 2010); they differ in their gating models that are constants rather than functions of the input. In Yang *et al.* (2009), the training set is divided into clusters as a preprocessing step and then cluster-specific kernel weights are learned using an alternating optimization method. The combined kernel function can be written as

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta_{c_i}^m k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m) \eta_{c_j}^m$$

where $\eta^m_{c_i}$ corresponds to the weight of kernel $k_m(\cdot,\cdot)$ in the cluster $\boldsymbol{x}_i$ belongs to. The kernel weights of the cluster that the test instance is assigned to are used in the testing phase. In Yang *et al.* (2010), instance-specific kernel weights are used instead of cluster-specific weights. The corresponding combined kernel function is

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P} \eta^m_i k_m(\boldsymbol{x}^m_i, \boldsymbol{x}^m_j)\eta^m_j$$

where η^m_i corresponds to the weight of kernel $k_m(\cdot,\cdot)$ for $\boldsymbol{x}_i$ and instance-specific weights are optimized using an alternating optimization problem for the training set. But, in the testing phase, the kernel weights for a test instance are all taken to be equal.

We are going to give our experimental results for classification and regression tasks on several data sets in Section 5.3.

4. LOCAL PROJECTION KERNELS

Dimensionality reduction is commonly used to alleviate the effect of redundant or correlated features and to visualize the training data using few, for example, two dimensions.

Principal Component Analysis (PCA) is the first method that comes to mind for linear dimensionality reduction (Pearson, 1901). PCA seeks to maximize the explained variance of the data in the projected feature space and performs a linear dimensionality reduction by calculating a projection matrix from the eigenvectors of the covariance matrix. It may perform badly for classification problems due to its linear and unsupervised nature. Kernel PCA (KPCA) is an extension to PCA algorithm that obtains nonlinear mappings with the help of kernel functions (Schölkopf and Smola, 2002).

Fisher Discriminant Analysis (FDA) is a well-known linear supervised method for dimensionality reduction that jointly minimizes the within-class variance and maximizes the between-class variance (Fisher, 1936). FDA has two main limitations: (a) the dimensionality of the projected feature space can be at most $K-1$ where K is the number of classes, and, (b) it assumes that each class follows a unimodal distribution, which may not always hold. FDA can also be kernelized to obtain nonlinear mappings.

Methods such as PCA and FDA learn a global projection matrix and use this matrix over the whole input space. This approach may not work for data sets that have a local neighborhood structure. A mixture of principal component analyzers has been proposed to capture regional differences in the covariance structure (Tipping and Bishop, 1999). The method divides the input density into clusters and learns a local PCA model in each cluster. However, the unsupervised nature of PCA method is preserved even though we learn local models.

Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Isomap (Tenenbaum *et al.*, 2000), and Laplacian Eigenmaps (Belkin and Niyogi, 2002) are some examples of unsupervised locality preserving manifold learning algorithms but these methods do not explicitly learn a mapping function for unseen data instances. Ocnlinx *et al.* (2009) and Hou *et al.* (2009) propose two variants of the LLE algorithm in order to capture the local neighborhood structure in the data better. Locality Preserving Projections (LPP) have been proposed also to learn a mapping function while preserving locality (He and Niyogi, 2004).

In addition to these unsupervised methods, there are also supervised methods that preserve the local neighborhood structure in the data. Local FDA (LFDA) combines the ideas behind FDA and LPP (Sugiyama, 2007). The mapping is obtained again by solving a generalized eigenvalue problem but the between-class scatter and within-class scatter matrices are calculated locally with the help of an affinity matrix (an idea that is borrowed from LPP). LFDA also removes the restriction of obtaining at most $K - 1$ dimensions in the projected feature space. Tao *et al.* (2009) extend FDA by maximizing the geometric mean of the divergences between different pairs of classes. This strategy obtains better projections in terms of class separation for multiclass problems. Another method is Local Learning Projections (LLP) that can use supervised information (its difference from PCA) and minimize the local estimation error instead of the global estimation error (Wu *et al.*, 2007).

Yan *et al.* (2007) formulate a common framework for representing different dimensionality reduction algorithms as graph embedding problems. For example, PCA, FDA, Isomap, LLE, LPP, and Laplacian Eigenmaps can be cast into a common formulation. Following this idea, a supervised variant of LLE called Discriminant LLE (DLLE) that also learns a mapping function is proposed (Li *et al.*, 2008).

In this chapter, we propose a supervised dimensionality reduction method coupled with a kernel machine called Local Projection Kernels (LPK) (Gönen and Alpaydın, 2010d). We reproduce the modification of the discriminant function of the SVM by integrating a projection matrix and explain how to optimize SVM parameters and the projection matrix jointly, as given by Chapelle *et al.* (2002). Then, we combine this idea of projections with the localized kernel idea of Chapter 3, and describe how to optimize all of the parameters in a coupled manner with an alternating optimization procedure.

4.1. Supervised Learning of Global Projection Kernels

Suppose that, instead of using the original features in the SVM formulation, we apply a linear projection to data instances with the projection matrix, $\mathbf{W} \in \mathbb{R}^{D \times R}$:

$$\boldsymbol{z} = \mathbf{W}^\top \boldsymbol{x}$$

where R is the dimensionality of the projected feature space. If we use the projected instances in the decision function, we obtain

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{z}) \rangle + b$$

and the primal problem becomes

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} \xi_i \\ \text{with respect to} \quad & \boldsymbol{w} \in \mathbb{R}^S, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}_+^N, \mathbf{W} \in \mathbb{R}^{D \times R} \\ \text{subject to} \quad & y_i \left(\langle \boldsymbol{w}, \Phi(\boldsymbol{z}_i) \rangle + b \right) \geq 1 - \xi_i \qquad \forall i. \end{aligned} \tag{4.1}$$

Note that the optimization problem in (4.1) is not convex due to the nonlinearity in the separation constraints.

Instead of trying to optimize the SVM parameters, $\{\boldsymbol{w}, b, \boldsymbol{\xi}\}$, and the projection matrix, $\mathbf{W}$, together, we utilize a two-step optimization algorithm as in Chapelle *et al.* (2002) and Rakotomamonjy *et al.* (2008). The algorithm starts with a random projection matrix. In the first step, we solve (4.1) with respect to $\{\boldsymbol{w}, b, \boldsymbol{\xi}\}$ while fixing $\mathbf{W}$. We then update $\mathbf{W}$ using a gradient-descent step calculated from the objective function of (4.1) in the second step. The following dual formulation can be solved instead of the primal formulation in the first step to apply the kernel trick:

$$\begin{aligned}
\text{maximize} \quad & J(\mathbf{W}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \underbrace{\langle \Phi(\boldsymbol{z}_i), \Phi(\boldsymbol{z}_j) \rangle}_{k(\boldsymbol{z}_i, \boldsymbol{z}_j)} \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_{+}^{N} \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \qquad \forall i.
\end{aligned} \tag{4.2}$$

For a fixed $\mathbf{W}$, we solve the dual optimization problem and obtain the optimal $\boldsymbol{\alpha}$ values. We need to update $\mathbf{W}$ by calculating the gradient of the objective function in (4.2). The gradient of the objective function with respect to the elements of $\mathbf{W}$ is calculated as

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}[k,l]} = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \frac{\partial k(\mathbf{W}^\top \boldsymbol{x}_i, \mathbf{W}^\top \boldsymbol{x}_j)}{\partial \mathbf{W}[k,l]} \qquad \forall (k,l)$$

where $k \in \{1, 2, \ldots, D\}$, $l \in \{1, 2, \ldots, R\}$, and $[\cdot, \cdot]$ indexes the elements of a matrix. The same gradient can also be obtained as the derivative of the margin (Chapelle *et al.*, 2002).

Three commonly used kernels, linear kernel (k_L), polynomial kernel (k_P), and Gaussian kernel (k_G), can be expressed in terms of $\mathbf{W}$ as follows:

$$\begin{aligned}
k_L(\boldsymbol{z}_i, \boldsymbol{z}_j) &= \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle = \boldsymbol{x}_i^\top \mathbf{W} \mathbf{W}^\top \boldsymbol{x}_j \\
k_P(\boldsymbol{z}_i, \boldsymbol{z}_j) &= (\langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle + 1)^q = (\boldsymbol{x}_i^\top \mathbf{W} \mathbf{W}^\top \boldsymbol{x}_j + 1)^q \\
k_G(\boldsymbol{z}_i, \boldsymbol{z}_j) &= \exp(-\, \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2^2 / \, s^2) \\
&= \exp(-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathbf{W} \mathbf{W}^\top (\boldsymbol{x}_i - \boldsymbol{x}_j) / \, s^2).
\end{aligned}$$

The derivatives of the kernels with respect to the elements of the projection matrix are given as

$$\begin{aligned}
\frac{\partial k_L(\boldsymbol{z}_i, \boldsymbol{z}_j)}{\partial \mathbf{W}[k,l]} &= \boldsymbol{x}_i[k]\boldsymbol{z}_j[l] + \boldsymbol{z}_i[l]\boldsymbol{x}_j[k] \\
\frac{\partial k_P(\boldsymbol{z}_i, \boldsymbol{z}_j)}{\partial \mathbf{W}[k,l]} &= (\boldsymbol{x}_i[k]\boldsymbol{z}_j[l] + \boldsymbol{z}_i[l]\boldsymbol{x}_j[k])q(\langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle + 1)^{q-1} \\
\frac{\partial k_G(\boldsymbol{z}_i, \boldsymbol{z}_j)}{\partial \mathbf{W}[k,l]} &= -2(\boldsymbol{x}_i[k] - \boldsymbol{x}_j[k])(\boldsymbol{z}_i[l] - \boldsymbol{z}_j[l])k_G(\boldsymbol{z}_i, \boldsymbol{z}_j)/s^2.
\end{aligned}$$

The projection matrix can be updated using a simple gradient-descent update rule with a fixed step size or Armijo's rule can be used to determine a better step size at each iteration. Note that this alternating optimization procedure does not guarantee convergence to the global optimum and the initial value of **W** may affect the solution quality. Figure 4.1 lists the main steps of the procedure that we call Global Projection Kernels (GPK).

```
1: Initialize $\mathbf{W}^{(0)}$ randomly
2: repeat
3:     Calculate $\mathbf{K}^{(t)} = \{k(\boldsymbol{z}_i, \boldsymbol{z}_j)\}_{i,j=1}^{N}$ using $\mathbf{W}^{(t)}$
4:     Solve kernel machine with $\mathbf{K}^{(t)}$
5:     Determine step size, $\mu^{(t)}$, using Armijo's rule
6:     Update projection matrix: $\mathbf{W}^{(t+1)} \Leftarrow \mathbf{W}^{(t)} - \mu^{(t)} \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
7: until convergence
```

Figure 4.1. Global Projection Kernels (GPK)

After convergence, we obtain the decision function in terms of model parameters as follows:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{W}^\top \boldsymbol{x}_i, \mathbf{W}^\top \boldsymbol{x}) + b.$$

We project both the input $\boldsymbol{x}$ and the support vector $\boldsymbol{x}_i$ to the lower dimensional space and calculate the kernel there.

4.2. Supervised Learning of Local Projection Kernels

Using a single projection matrix over the whole input space can not capture multiple modalities that may exist in the data. At this point, we can combine localized kernel functions of Chapter 3 with projection matrices and similar to using kernel functions with changing weights in different regions, we can divide the input space into P regions and learn a local projection matrix, $\mathbf{W}_m \in \mathbb{R}^{D_m \times R_m}$, $m = 1, \dots, P$, in each region, in order to capture the local structure information. So, we have P different projected data instances for each instance:

$$\boldsymbol{z}^m = \mathbf{W}_m^\top \boldsymbol{x}^m \qquad \forall m$$

and the discriminant function can be rewritten as

$$f(\boldsymbol{x}) = \sum_{m=1}^{P} \eta_m(\boldsymbol{x}|\mathbf{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{z}^m) \rangle + b$$

where the gating model, $\eta_m(\boldsymbol{x}|\mathbf{V})$, now chooses the weight for the corresponding projected feature space, $\Phi_m(\boldsymbol{z}^m)$, as a function of input $\boldsymbol{x}$. By modifying the formulation in (4.1) with this new discriminant function, we get the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \sum_{m=1}^{P} \|\boldsymbol{w}_m\|_2^2 + C \sum_{i=1}^{N} \xi_i \\ \text{with respect to} \quad & \boldsymbol{w}_m \in \mathbb{R}^{S_m}, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}_+^N, \mathbf{V} \in \mathbb{R}^{P \times (D_\mathcal{G}+1)}, \mathbf{W}_m \in \mathbb{R}^{D_m \times R_m} \\ \text{subject to} \quad & y_i \left(\sum_{m=1}^{P} \eta_m(\boldsymbol{x}_i|\mathbf{V}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{z}_i^m) \rangle + b \right) \geq 1 - \xi_i \qquad \forall i \end{aligned} \tag{4.3}$$

and this problem is not convex, either.For given $\mathbf{V}$ and $\{\mathbf{W}_m\}_{m=1}^P$ values, (4.3) becomes convex and we can obtain the Lagrangian of the primal problem:

$$\begin{aligned} L_D(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P) = & \frac{1}{2}\sum_{m=1}^{P} \|\boldsymbol{w}_m\|_2^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\beta_i\xi_i \\ & - \sum_{i=1}^{N}\alpha_i\left(y_i\left(\sum_{m=1}^{P}\eta_m(\boldsymbol{x}_i|\mathbf{V})\langle\boldsymbol{w}_m, \Phi_m(\boldsymbol{z}_i^m)\rangle + b\right) - 1 + \xi_i\right) \end{aligned}$$

and taking the derivatives of L_D with respect to the primal variables gives

$$\begin{aligned} &\frac{\partial L_D(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \boldsymbol{w}_m} \Rightarrow \boldsymbol{w}_m = \sum_{i=1}^{N}\alpha_i y_i \eta_m(\boldsymbol{x}_i|\mathbf{V})\Phi_m(\boldsymbol{z}_i^m) \qquad \forall m \\ &\frac{\partial L_D(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial b} \Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0 \\ &\frac{\partial L_D(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \xi_i} \Rightarrow C = \alpha_i + \beta_i \qquad \forall i. \end{aligned} \tag{4.4}$$

From $L_D(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)$ and (4.4), the dual formulation is obtained as

$$\begin{aligned} \text{maximize} \quad & J(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^N \\ \text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \qquad \forall i \end{aligned} \tag{4.5}$$

where the *local projection kernel function* is defined as

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{P}\eta_m(\boldsymbol{x}_i|\mathbf{V})\underbrace{\langle\Phi_m(\boldsymbol{z}_i^m), \Phi_m(\boldsymbol{z}_j^m)\rangle}_{k_m(z_i^m, z_j^m)}\eta_m(\boldsymbol{x}_j|\mathbf{V})$$

and using $k_{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ corresponds to projecting data instances into the $(\sum_{m=1}^{P} S_m)$-dimensional feature space and using the dot product in this feature space.

$$\begin{pmatrix} \eta_1(\boldsymbol{x}_j|\mathbf{V})\Phi_1(\mathbf{W}_1^\top \boldsymbol{x}_i^1) \\ \eta_2(\boldsymbol{x}_j|\mathbf{V})\Phi_2(\mathbf{W}_2^\top \boldsymbol{x}_i^2) \\ \vdots \\ \eta_P(\boldsymbol{x}_j|\mathbf{V})\Phi_P(\mathbf{W}_P^\top \boldsymbol{x}_i^P) \end{pmatrix}^\top \begin{pmatrix} \eta_1(\boldsymbol{x}_j|\mathbf{V})\Phi_1(\mathbf{W}_1^\top \boldsymbol{x}_j^1) \\ \eta_2(\boldsymbol{x}_j|\mathbf{V})\Phi_2(\mathbf{W}_2^\top \boldsymbol{x}_j^2) \\ \vdots \\ \eta_P(\boldsymbol{x}_j|\mathbf{V})\Phi_P(\mathbf{W}_P^\top \boldsymbol{x}_j^P) \end{pmatrix}$$

Having fixed the SVM parameters and the gating model parameters, we can update the local projection matrices using gradient-descent. For given $\boldsymbol{\alpha}$ and $\mathbf{V}$ values, the gradient of the objective function in (4.5) with respect to the elements of $\mathbf{W}_m$ matrices is given as

$$\frac{\partial J(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \mathbf{W}_m[k,l]} = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \Upsilon_i^j \eta_m(\boldsymbol{x}_i|\mathbf{V}) \frac{\partial k_m(\boldsymbol{z}_i^m, \boldsymbol{z}_j^m)}{\partial \mathbf{W}_m[k,l]} \eta_m(\boldsymbol{x}_j|\mathbf{V}).$$

Having fixed the SVM parameters and the local projection matrices, we can update the gating model parameters. For given $\boldsymbol{\alpha}$ and $\{\mathbf{W}_m\}_{m=1}^P$ values, the gradients of the objective function in (4.5) with respect to the parameters of the softmax gating model (3.2) are given as

$$\begin{aligned} \frac{\partial J(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \boldsymbol{v}_m} &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{h=1}^{P} \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V}) k_h(\boldsymbol{z}_i^h, \boldsymbol{z}_j^h) \eta_h(\boldsymbol{x}_j|\mathbf{V}) \\ &\quad (\boldsymbol{x}_i^{\mathcal{G}}(\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V})) + \boldsymbol{x}_j^{\mathcal{G}}(\delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V}))) \\ \frac{\partial J(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial v_{m0}} &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{h=1}^{P} \Upsilon_i^j \eta_h(\boldsymbol{x}_i|\mathbf{V}) k_h(\boldsymbol{z}_i^h, \boldsymbol{z}_j^h) \eta_h(\boldsymbol{x}_j|\mathbf{V}) \\ &\quad (\delta_m^h - \eta_m(\boldsymbol{x}_i|\mathbf{V}) + \delta_m^h - \eta_m(\boldsymbol{x}_j|\mathbf{V})). \end{aligned}$$

The complete algorithm of our proposed LPK is summarized in Figure 4.2. The convergence of the algorithm can be determined by observing the change in the objective function value of (4.5).

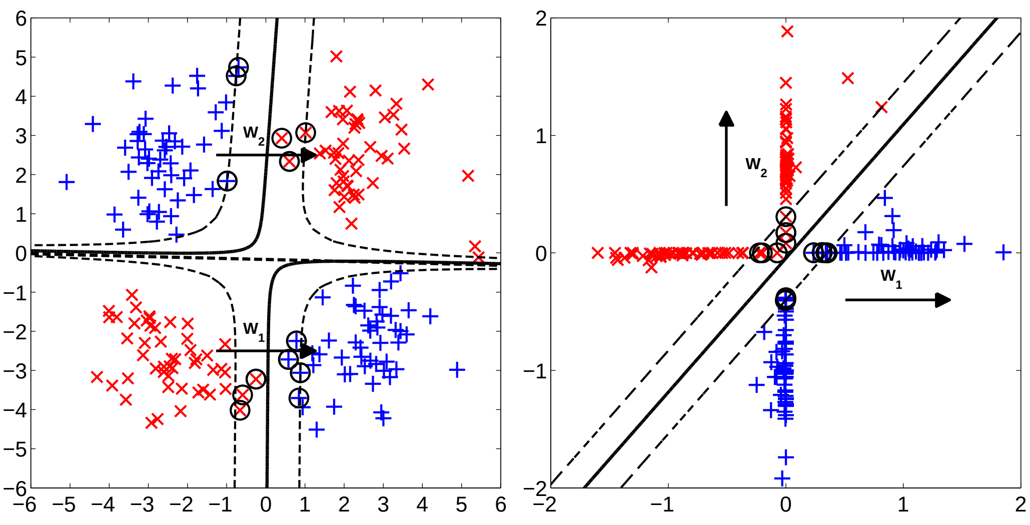
```
1: Initialize $\mathbf{V}^{(0)}$ and $\{\mathbf{W}_m^{(0)}\}_{m=1}^P$ randomly
2: **repeat**
3:    Calculate $\mathbf{K}_\eta^{(t)} = \{k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i,j=1}^N$ using $\mathbf{V}^{(t)}$ and $\{\mathbf{W}_m^{(t)}\}_{m=1}^P$
4:    Solve kernel machine with $\mathbf{K}_\eta^{(t)}$
5:    Determine step size, $\mu^{(t)}$, using Armijo's rule
6:    Update local projection matrices: $\mathbf{W}_m^{(t+1)} \Leftarrow \mathbf{W}_m^{(t)} - \mu^{(t)} \dfrac{\partial J(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \mathbf{W}_m}$    $\forall m$
7:    Calculate $\mathbf{K}_\eta^{(t)} = \{k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i,j=1}^N$ using $\mathbf{V}^{(t)}$ and $\{\mathbf{W}_m^{(t+1)}\}_{m=1}^P$
8:    Solve kernel machine with $\mathbf{K}_\eta^{(t)}$
9:    Determine step size, $\Delta^{(t)}$, using Armijo's rule
10:    Update gating model parameters: $\mathbf{V}^{(t+1)} \Leftarrow \mathbf{V}^{(t)} - \Delta^{(t)} \dfrac{\partial J(\mathbf{V}, \{\mathbf{W}_m\}_{m=1}^P)}{\partial \mathbf{V}}$
11: **until** convergence
```

Figure 4.2. Local Projection Kernels (LPK)

After determining the final $\boldsymbol{\alpha}$, b, $\mathbf{V}$, and $\{\mathbf{W}_m\}_{m=1}^P$ values, the resulting discriminant function is:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \sum_{m=1}^{P} \alpha_i y_i \eta_m(\boldsymbol{x}_i|\mathbf{V}) k_m(\mathbf{W}_m^\top \boldsymbol{x}_i^m, \mathbf{W}_m^\top \boldsymbol{x}^m) \eta_m(\boldsymbol{x}|\mathbf{V}) + b.$$

In order to better illustrate our proposed method, we create a toy data set consisting of four clusters (two for each class) as shown in Figure 4.3(a). If we use a global projection matrix over the whole input space, we can not obtain a clear linear separation between classes due to intraclass multimodalities. However, we can obtain a projected space in which classes are well-separated and multimodal structures in each class are preserved, as shown in Figure 4.3(b), by splitting the input space into two regions using the softmax gating model (shown with the thick dashed line in Figure 4.3(a)) and performing local projections (one-dimensional projections, $\mathbf{W}_1 \in \mathbb{R}^{2\times 1}$ and $\mathbf{W}_2 \in \mathbb{R}^{2\times 1}$, shown with arrows in Figure 4.3(a)) in each region.



(a) Original Feature Space (b) Projected Feature Space

Figure 4.3. Motivating example for learning local projections. The solid lines show the resulting discriminants in the original and projected feature spaces. (a) There are two local regions in the original feature space and the thick dashed line separate them. $\mathbf{W}_1$ and $\mathbf{W}_2$ arrows show the projection directions in the two regions. (b) The horizontal and vertical axes correspond to the projected directions in the two regions.

Figure 4.4 shows the gating model output for the first projection matrix, $\eta_1(\boldsymbol{x}|\mathbf{V})$, superimposed with training data. We see that $\eta_1(\boldsymbol{x}|\mathbf{V})$ divides the input space into two local regions. The line where $\eta_1(\boldsymbol{x}|\mathbf{V}) = 0.5$ is shown by a thick dashed line in Figure 4.3(a).

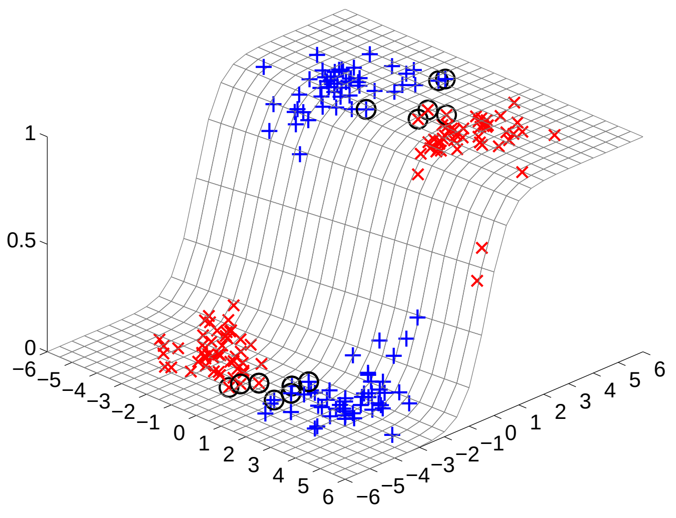


Figure 4.4. The gating model output superimposed with training data for the motivating example.

4.3. Discussion

In LPK training, the gradient calculations have ignorable time complexity compared with the SVM solver and these calculations are made by using only the support vectors at the current iteration. The key issue for faster convergence is to select good gradient-descent step sizes, ($\Delta^{(t)}$ and $\mu^{(t)}$ in Figure 4.2), at each iteration. Better step size values can be obtained by utilizing a line search method such as Armijo's rule but this process needs additional calls to the SVM solver. Clearly, the time complexity for each iteration increases but the algorithm converges in fewer iterations. In our experiments, we use Armijo's rule to determine the step sizes at each iteration and the algorithm converges in a few iterations (generally five to 10). A more detailed convergence analysis is performed in Subsection 5.4.4.

We describe LPK for binary classification problems but the same idea can easily be applied to regression estimation and novelty detection problems (Schölkopf and Smola, 2002) by changing the dual optimization problem (4.5) solved at each iteration and calculating the gradients with respect to the new objective function. The gradient formulations obtained for binary classification problems can be used by just replacing Υ_i^j with $(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)$ for SVR and with $\alpha_i \alpha_j$ for OCSVM.

Using local projection matrices in different regions of the input space enables us to extract information about the relative importance of features in each region. The features with high magnitude weights in local projection matrices give more information in the corresponding region of the input space. The features with very small weights can also be discarded to perform feature selection locally.

Coupled learning of a data projection rule and a classification algorithm has also been studied by Weinberger and Saul (2009) and Globerson and Roweis (2006). In these studies, a Mahalanobis distance metric used in nearest neighbor classification is learned by directly considering the classification accuracy. Tao *et al.* (2005a) propose a supervised learning method that performs learning and feature extraction together for tensor data. The discriminant parameters and the projection matrix are optimized using an alternating approach. Our proposed LPK is more similar to Pereira and Gordon (2006) in that the optimization of the projection matrix and the classifier (SVM as in our case) performed jointly. They use a global projection matrix over the whole input space, but we introduce a data-dependent projection by using a gating model for choosing the projection matrix.

We are going to give our experimental results for visualization and classification tasks in Section 5.3.

5. EXPERIMENTS

This chapter gives experimental results that support utility of our proposed methods on benchmark data sets from the UCI Machine Learning Repository as well as several image image recognition and bioinformatics data sets.

5.1. Methodology

Except otherwise stated, our experimental methodology is as follows: Given a data set, if separate training and test splits are not supplied, a random one-third is reserved as the test set and the remaining two-thirds is resampled using 5×2 cross-validation to generate 10 training and validation sets, with stratification for classification problems. The validation sets of all folds are used to optimize C and for regression problems, ϵ, the width of the error tube. The best configuration (measured as the highest average classification accuracy or the lowest mean square error (MSE) for regression problems) on the validation folds is used to train the final classifiers/regressors on the training folds and their performance is measured over the test set. So, for each data set, we have ten test set results; we report their averages and one standard deviations.

We use our own implementations of SVM, SVR, MKL, RMKL, LMKL, GPK, and LPK written in MATLAB and the resulting optimization problems for all these methods are solved using the MOSEK optimization software (Mosek, 2010). We stop the algorithms of LMKL, GPK, and LPK when the objective function value of the current iteration is not less than $(1 - \tau)$ times the objective function value of the previous iteration. The parameter τ is set to 0.001 in our experiments.

5.2. Multiple Kernel Learning Experiments

This section lists the results of intermediate integration experiments and compares MKL with its regularized and cost-conscious variants we propose.

5.2.1. Intermediate Integration Experiments

In protein design and analysis, understanding the stability in sequence, structure, and function paradigms is of importance (Lee and Levitt, 1991) and hence there is a need for predicting the protein stability change due to mutation. Single amino acid mutations can significantly change the stability of a protein structure (Cheng *et al.*, 2006). To acquire a set of experimental annotations for every possible random mutation is combinatorial and requires significant resources and time. Thus, accurate computational prediction would be of use for suggesting the destructive mutations as well as the most favorable and stable novel protein sequences. To this end, the prediction of protein stability change due to amino acid substitutions remains a challenging task in the field of molecular biology.

One can predict the direction towards which the mutation shifts the stability of the protein (namely the sign of $\Delta\Delta$G). It could be positive or negative, corresponding to an increase or decrease in stability, respectively. From a machine learning perspective, this is a binary classification task, where given $\boldsymbol{x}$, information about the single-site amino acid substitution, the aim is to decide whether this is a positive or negative example, depending on whether the mutation is favorable or not.

5.2.1.1. Data Set. We extract a data set (S2783) that contains 2783 single-site mutations with known PROTEIN DATA BANK (PDB) code of the protein and $\Delta\Delta$G values from the ProTherm database (Gromiha *et al.*, 2000).[2] Each instance has the following features: PDB code of the protein, mutated position and mutation itself, solvent accessibility (SA), pH value, temperature (T), and the change in the free energy ($\Delta\Delta$G) due to a mutation in a single position. As there are instances for the same mutation and position where $\Delta\Delta$G differs with T and pH values, T and pH are kept as features in our data set. After removing the instances with missing values, S2783 reduces to 2471 instances from 68 different proteins and 755 of them (30.55 per cent) are stabilizing mutations.

[2]Experiments done in Subsection 5.2.1 are the results of joint work with Ayşegül Özen and Türkan Haliloğlu from the Deparment of Chemical Engineering.

5.2.1.2. Added Features. The substitution frequency of an amino acid for another is considered here as an additional feature with the Point Accepted Mutation (PAM) matrix (Dayhoff *et al.*, 1978). PAM250 is chosen for the score of each amino acid substitution and is based on the frequency of that substitution in closely related proteins that have experienced a certain amount of evolutionary divergence.

Another feature considered is the mobility/flexibility of the amino acid position in a given structure. The B-Factors reported in the PDB file is a good and quick indicator of this feature. Neighbors of the mutated residue in both amino acid sequence and 3D structure are the two other features that have been used recently (Capriotti *et al.*, 2004; Cheng *et al.*, 2006). A window size of seven in the sequence (Cheng *et al.*, 2006) and a cutoff distance of $9\mathring{A}$ in space (Capriotti *et al.*, 2004) are previously used to find the neighbors of the mutated position as the optimum sequence length and distance. In our implementation, in addition to alpha-carbon atoms (C_α), beta-carbon (C_β) atoms are also considered to reflect the packing at a relatively higher resolution.

A mutation in a position of a protein sequence will change the number of side-chain atoms of the residue in that position. This may trigger a conformational change or local readjustments that may result also in a change in the atomic packing around that residue and the fluctuations of the surrounding residues and the mutated residue itself. Nevertheless, as in other studies (Capriotti *et al.*, 2004; Cheng *et al.*, 2006; Huang *et al.*, 2006), we neglect this effect.

Table 5.1 gives a list of the representations, original features, and the new features that we introduce. The information coming only from the sequence (SO), and the topology of the protein structure (TO), and both (ST) are encoded in the same way as defined in previous studies (Cheng *et al.*, 2006). An added asterisk, for example, (SO*), denotes the representation with newly added features. Neighbors of the mutated position in the sequence, mutation, T, and pH are encoded in SO/SO*. Sequence information is not used in TO/TO*; instead, spatial neighbors and the solvent accessibility of the mutated position are encoded. In ST/ST*, all information are combined. The substitution likelihood of an amino acid is added to the existing data as a new feature in all three representations. Crystallographic B-FACTORs of the C_α and C_β atoms are used in TO* and ST*. For discrete features like amino acid identities, *1-of-n encoding* is used, that is, if the variable can take one of n different values, one is set to 1 and all others to 0.

Table 5.1. Representations, original features, and the new features of the S2783 data set. In all three representations, amino acid substitution likelihood is used as a feature. B-FACTORs of the C_α and C_β atoms and spatial neighbor determined using both C_α and C_β atoms are features introduced into TO and ST.

Representation	Original Features	New Representation	New Features
SO	±3 Neighbors (±3NE)	SO*	PAM250 (PAM)
	Mutation (MUT)		
	T/PH		
TO	Mutation (MUT)	TO*	PAM250 (PAM)
	C_α Contacts (CA)		C_α B-FACTORs (BFA)
	SA/T/PH		C_β B-FACTORs (BFB)
			C_α and C_β Contacts (CB)
ST	±3 neighbors (±3 NE)	ST*	PAM250 (PAM)
	Mutation (MUT)		C_α B-FACTORs (BFA)
	C_α Contacts (CA)		C_β B-FACTORs (BFB)
	SA/T/PH		C_α and C_β Contacts (CB)

5.2.1.3. Methodology. We use a slightly different methodology on the S2783. We apply 20-fold cross-validation on the training set and obtain 20 folds. This whole process is replicated 10 times each time using a different random test set. As a result, we obtain 10×20 test set results and report the average of these results.

5.2.1.4. Results. We perform experiments with early, late, and intermediate integration using both classification and regression formulation with/without the reject option (Özen *et al.*, 2009). But, we report here only the results of intermediate integration using classification formulation without the reject option. The average test accuracies for all data representations using intermediate integration are given in Table 5.2. When we use MKL (linear formulation of (Bach *et al.*, 2004)), we can see that adding extra features does not change accuracy. The highest accuracy is obtained with ST as 0.806.

Table 5.2. The average test accuracies using intermediate integration on the S2783 data set.

SO	SO*	TO	TO*	ST	ST*
0.800	0.799	0.805	0.802	0.806	0.804

When we look at Table 5.3, we can say that the added features carry information for predicting the energy change for single-site mutations even though they do not improve the average testing accuracy. Local spatial composition with C_α and C_β (CB) has larger weight than local spatial composition with C_α (CA) and the information that reflects the extents of mobility/flexibility of C_α and C_β (BFA and BFB) has nonzero weights.

Table 5.3. The combination weights obtained using intermediate integration for the original and modified feature sets on the S2783 data set.

SO	(0.19)1NE + (0.20)2NE + (0.23)3NE + (0.27)MUT + (0.09)T + (0.03)PH
SO*	(0.19)1NE + (0.20)2NE + (0.22)3NE + (0.27)MUT + (0.09)T + (0.03)PH + (0.00)PAM
TO	(0.19)MUT + (0.56)CA + (0.17)SA + (0.05)T + (0.02)PH
TO*	(0.21)MUT + (0.23)CA + (0.12)SA + (0.06)T + (0.02)PH + (0.00)PAM + (0.23)CB + (0.07)BFA + (0.06)BFB
ST	(0.04)1NE + (0.03)2NE + (0.04)3NE + (0.21)MUT + (0.45)CA + (0.15)SA + (0.06)T + (0.02)PH
ST*	(0.02)1NE + (0.02)2NE + (0.03)3NE + (0.21)MUT + (0.21)CA + (0.11)SA + (0.06)T + (0.02)PH + (0.00)PAM + (0.19)CB + (0.06)BFA + (0.06)BFB

5.2.2. Regularized Multiple Kernel Learning Experiments

In this section, we perform experiments using the regularized MKL approach of Section 2.7. In this set of experiments, C parameter is selected from {0.01, 0.1, 1, 10, 100} using cross-validation. Δ parameter for RSM method is selected as 0.3 in our experiments. This corresponds to selecting the low and high factor levels as 0.5

($10^{-0.3}$) and 2 ($10^{+0.3}$). In the result tables, we report the average test accuracies, support vector percentages, combination weights, and regularization parameters. The average test accuracies and support vector percentages are made bold if the difference between MKL and the regularized variant is statistically significant using the 5×2 cv paired F test (Alpaydın, 1999). We also report the count of (W)ins-(T)ies-(L)osses of kernel combination with RMKL from direct comparison and the 5×2 cv paired F test. The Wilcoxon's signed-rank test is used to compare the two variants over a number of data sets in terms of average accuracies and support vector percentages and the result is shown as (W)in, (T)ie, or (L)oss (Wilcoxon, 1945). For both statistical tests, the significance level, α, is taken as 0.05.

5.2.2.1. Combining General Purpose Kernels. We perform experiments on five different bioinformatics data sets by combining the linear kernel, the second-degree polynomial kernel and the Gaussian kernel whose width parameter is estimated as in (2.10). ACCEPTORS and DONORS are human splice site detection data sets consisting of 3889 and 6246 data instances, respectively (Kulp *et al.*, 1996). ARABIDOPSIS and VERTEBRATES are translation initiation site detection data sets containing 2048 and 13454 instances, respectively (Pedersen and Nielsen, 1997). POLYADENYLATION is a polyadenylation signal prediction data set containing 9255 instances for human DNA and mRNA sequences (Liu *et al.*, 2003). We use the supplied training and test splits on ACCEPTORS, DONORS, and POLYADENYLATION data sets.

We see in Table 5.4 that RMKL uses statistically significantly fewer support vectors on ACCEPTORS and DONORS data sets and obtains statistically significantly higher accuracy on the ACCEPTORS data set. MKL uses k_L and k_P with nonzero weights, whereas RMKL assigns zero weights to these kernels and uses only k_G on the ACCEPTORS data set. This selection improves the average test accuracy statistically significantly and even though RMKL uses only the Gaussian kernel, it stores statistically significantly fewer support vectors. On the DONORS data set, RMKL assigns more weight to the linear kernel, removing the second-order polynomial kernel and using the Gaussian kernel less; the accuracy does not change but the percentage of support vectors decreases statistically significantly. This shows that RMKL is able to choose between kernels depending on the complexity of the discriminant to be learned: If the boundary needs be complex, the Gaussian kernel is favored; if it is simple, linear or polynomial kernels are given higher weights.

Table 5.4. The average test accuracies, support vector percentages, combination weights, and regularization parameters with (k_L-k_P-k_G) combination on bioinformatics data sets.

	MKL		RMKL		
	Test Acc.	SV	Test Acc.	SV	
	η_L-η_P-η_G		η_L-η_P-η_G		d_L-d_P-d_G
ACCEPTORS	90.45±0.42	50.12±1.11	**91.19±0.38**	**44.90±0.82**	
	0.30-0.70-0.00		0.00-0.00-4.59		1.00-1.55-0.47
DONORS	94.77±0.28	27.78±0.47	94.78±0.21	**26.25±0.45**	
	0.11-0.03-0.86		0.17-0.00-0.73		1.00-1.52-1.06
ARABIDOPSIS	85.29±0.84	62.32±1.10	**85.92±0.85**	62.39±0.98	
	0.25-0.75-0.00		0.00-0.00-5.16		1.00-1.00-0.44
VERTEBRATES	86.22±0.22	53.84±0.72	86.15±0.30	**40.86±0.44**	
	0.20-0.80-0.00		0.70-0.15-0.00		1.00-1.42-1.16
POLYADENYLATION	68.25±1.24	72.14±1.02	68.70±1.34	72.81±1.06	
	0.01-0.16-0.84		0.00-0.00-1.09		1.00-1.54-0.96

On the ARABIDOPSIS data set, too, RMKL chooses to use only k_G instead of using both k_L and k_P. It obtains statistically significantly higher accuracy by storing a comparable number of support vectors. On the VERTEBRATES data set, RMKL gives higher weight to k_L than k_P and this causes a significant decrease in the support vector count. RMKL stores nearly 13 per cent fewer support vectors than MKL.

On the POLYADENYLATION data set, RMKL obtains comparable accuracy and support vector count. k_L and k_P are replaced with k_G (one kernel is calculated instead of two) and RMKL increases the average test accuracy but not statistically significantly.

We also perform experiments on the Multiple Features (MULTIFEAT) digit recognition data set from the UCI Machine Learning Repository, composed of six different data representations for 2000 handwritten numerals. The properties of these feature representations are summarized in Table 5.5. Two binary classification problems are generated from the MultiFeat data set: In the MultiFeat-EO data set, we separate even digits from odd digits; in the MultiFeat-SL data set, we separate small (‘0’ - ‘4’) digits from large (‘5’ - ‘9’) digits.

Table 5.5. Multiple feature representations in the MultiFeat data set.

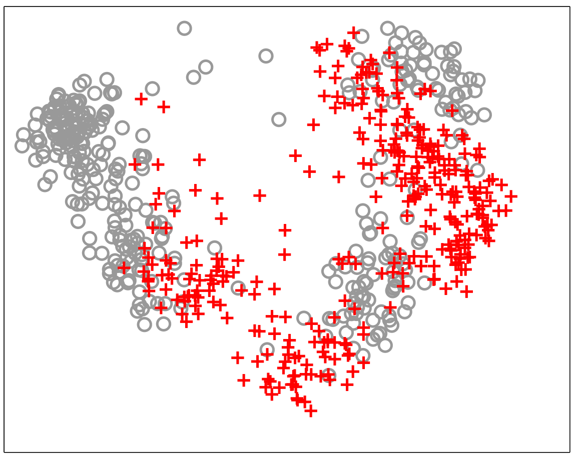
Name	Dimension	Data Source
Fac	216	Profile correlations
Fou	76	Fourier coefficients of the shapes
Kar	64	Karhunen-Loève coefficients
Mor	6	Morphological features
Pix	240	Pixel averages in 2×3 windows
Zer	47	Zernike moments

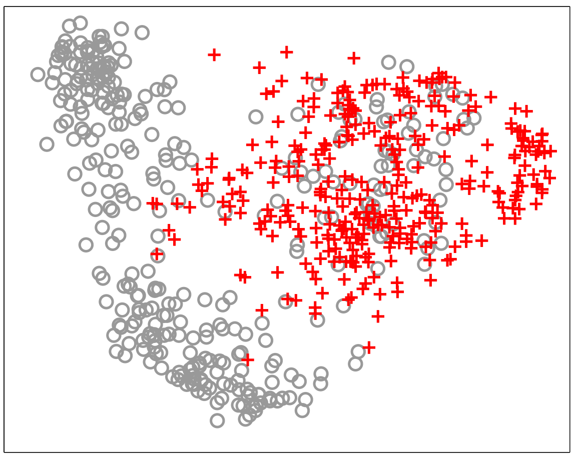
In our experiments, we combine six linear kernels calculated on each of the representations by MKL and RMKL. We see in Table 5.6 that RMKL uses five data representations (eliminating Pix by assigning zero weight) on the MultiFeat-EO data set, whereas MKL uses all data representations with nonzero weights. Both methods obtain comparable average accuracy results on the test set. On the MultiFeat-SL data set, RMKL eliminates two data representations (Pix and Zer) without a significant decrease in accuracy while storing statistically significantly fewer support vectors.

Table 5.6. The average test accuracies, support vector percentages, combination weights, and regularization parameters with (k_{Fac}-k_{Fou}-k_{Kar}-k_{Mor}-k_{Pix}-k_{Zer}) combination on the MultiFeat data set.

	MKL		RMKL		
	Test Acc.	SV	Test Acc.	SV	
	η_1-η_2-η_3-η_4-η_5-η_6		η_1-η_2-η_3-η_4-η_5-η_6		d_1-d_2-d_3-d_4-d_5-d_6
EO	98.31±0.34	14.86±0.79	98.18±0.29	14.50±0.96	
	0.30-0.29-0.11-0.01-0.28-0.02		0.40-0.24-0.22-0.05-0.00-0.27		1.00-1.07-1.02-0.58-1.63-0.53
SL	97.40±0.37	32.59±0.82	97.13±0.31	**27.58±1.56**	
	0.25-0.30-0.09-0.15-0.15-0.07		0.31-1.11-0.13-0.47-0.00-0.00		1.00-0.59-1.19-0.50-1.58-2.14

In order to see the differences between kernel combination rules, we apply KPCA to the kernels obtained using unweighted sum (SUM), MKL, and RMKL. Figure 5.1 gives the two-dimensional projections obtained on the MULTIFEAT-EO data set. We see that using a weighted sum (learning the combination weights using MKL or RMKL) produces a better projection than using an unweighted sum; there does not seem to be a significant difference between the projections obtained by MKL and RMKL.


(a) SUM


(b) MKL

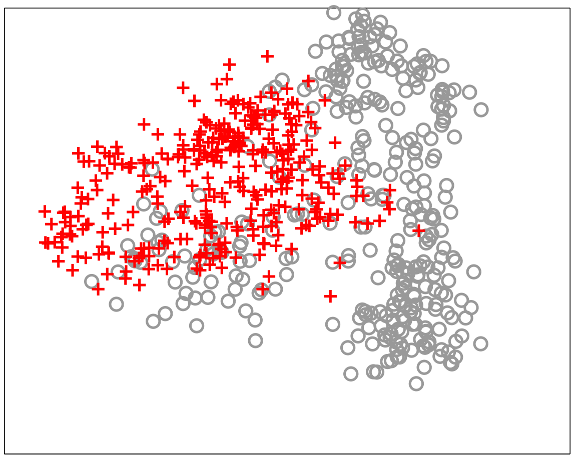

(c) RMKL

Figure 5.1. Two-dimensional projections obtained on the MULTIFEAT-EO data set by KPCA. (a) SUM: The first two eigenvectors explains 24.86 per cent of the variance. (b) MKL: The first two eigenvectors explains 23.66 per cent of the variance. (c) RMKL: The first two eigenvectors explains 16.32 per cent of the variance.

5.2.2.2. Combining Domain-Specific Kernels. We perform protein function prediction experiments on the MIPS Comprehensive Yeast Genome Database (CYGD) (Mewes *et al.*, 2000) that categorizes 3588 proteins into 13 top-level categories that can be interpreted as 13 binary classification tasks (Y1, Y2, . . . , Y13), one for each category. The reason for decomposing into binary classification problems instead of using a multiclass formulation is that some proteins belong to more than one category. We use the eight kernel functions shown in Table 5.7, also used in Lanckriet *et al.* (2004b).

Table 5.7. Kernels for protein function prediction problem (Lanckriet *et al.*, 2004b).

Kernel	Explanation	Data Source
k_{Pfam}	Pfam kernel	Protein sequences
k_{PfamE}	Enriched Pfam kernel	Protein sequences
k_{TAP}	Diffusion kernel	Protein interactions
k_{Phys}	Diffusion kernel	Protein interactions
k_{Gen}	Diffusion kernel	Protein interactions
k_{Exp}	Correlation kernel	Gene expression profiles
k_{ExpG}	Gaussian kernel	Gene expression profiles
k_{SW}	Smith-Waterman kernel	Protein sequences

Two different kernel subsets described in Lanckriet *et al.* (2004b) are also used in our experiments: (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) and (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}).

In Table 5.8, comparing MKL and our proposed regularized variant, RMKL, we see that the average accuracy percentage on the test set remains statistically similar on the 13 tasks for the first subset. Direct comparison between average accuracies shows that the average accuracy increases on 10 out of 13 tasks, and the Wilcoxon's signed-rank test finds a significant win of RMKL accuracy over MKL. Comparing support vector percentages, RMKL has six significant wins and eight direct comparison wins; using the Wilcoxon's signed-rank test however, there is no significant difference between the support vector percentages.

We can also compare the combination weights ($\{\eta_m\}_{m=1}^{P}$) for both methods and the regularization parameters ($\{d_m\}_{m=1}^{P}$) found by RMKL in Table 5.8, and we see

Table 5.8. The average test accuracies, support vector percentages, combination weights, and regularization parameters with (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) combination on the protein function prediction experiments.

	MKL		RMKL		
	Test Acc.	SV	Test Acc.	SV	
	η_1-η_2-η_3-η_4-η_5		η_1-η_2-η_3-η_4-η_5		d_1-d_2-d_3-d_4-d_5
Y1	78.59±0.87	**91.50±1.40**	79.21±0.45	94.85±1.05	
	0.70-0.04-0.08-0.14-0.05		0.46-0.06-0.05-0.12-0.09		1.00-1.23-1.32-1.30-1.33
Y2	93.23±0.00	90.95±3.15	93.56±0.25	85.93±2.51	
	0.37-0.03-0.15-0.22-0.22		0.34-0.25-0.41-0.00-0.00		1.00-1.00-1.00-2.06-2.06
Y3	87.17±0.39	96.11±0.40	87.47±0.28	**93.14±1.05**	
	0.12-0.02-0.18-0.32-0.37		0.23-0.07-0.25-0.45-0.00		1.00-1.00-1.00-1.00-2.08
Y4	85.62±0.51	87.22±1.43	87.07±0.58	**82.81±2.07**	
	0.22-0.03-0.24-0.30-0.21		0.34-0.22-0.33-0.04-0.00		1.00-0.94-1.10-1.33-1.57
Y5	91.90±0.26	91.24±1.44	92.29±0.21	**77.52±4.48**	
	0.20-0.01-0.18-0.26-0.34		0.35-0.01-0.00-0.00-0.74		1.00-1.25-1.39-1.59-0.93
Y6	86.73±0.36	93.58±1.26	87.64±0.43	92.78±1.26	
	0.59-0.00-0.15-0.25-0.01		0.40-0.01-0.04-0.46-0.00		1.00-1.24-1.32-1.06-1.76
Y7	90.31±0.30	**82.56±0.88**	90.49±0.36	86.45±0.68	
	0.20-0.00-0.28-0.22-0.30		0.32-0.00-0.54-0.10-0.00		1.00-1.37-0.97-1.30-1.62
Y8	92.92±0.20	91.49±3.03	93.26±0.26	95.91±1.34	
	0.16-0.02-0.16-0.22-0.45		1.00-0.00-0.00-0.00-0.00		1.00-10.0-10.0-10.0-10.0
Y9	94.75±0.10	82.28±3.69	95.20±0.40	84.18±2.39	
	0.14-0.15-0.19-0.10-0.42		0.39-0.00-0.12-0.00-0.10		1.00-2.39-1.54-3.18-1.78
Y10	89.16±0.36	93.39±1.51	89.05±0.32	**75.94±2.87**	
	0.09-0.00-0.27-0.24-0.39		0.23-0.00-0.25-0.45-0.00		1.00-1.41-1.13-1.00-1.42
Y11	94.65±0.00	96.24±1.05	94.43±0.13	**86.68±1.79**	
	0.05-0.03-0.15-0.17-0.60		0.22-0.15-0.24-0.39-0.00		1.00-1.00-1.00-1.00-2.08
Y12	95.26±0.33	76.32±1.91	95.24±0.45	79.63±2.24	
	0.99-0.01-0.00-0.00-0.00		0.57-0.01-0.05-0.20-0.00		1.00-1.37-1.33-1.27-1.33
Y13	97.73±0.04	77.14±4.49	97.74±0.00	**9.74±0.83**	
	0.11-0.12-0.15-0.18-0.44		0.13-0.15-0.20-0.00-0.52		1.00-1.00-1.00-1.41-1.00
5×2 cv Paired F Test (W-T-L)			0-13-0	6-5-2	
Direct Comparison (W-T-L)			10-0-3	8-0-5	
Wilcoxon's Rank Test (W/T/L)			W	T	

that another difference between these two methods is the number of active kernels with nonzero weights. RMKL method, on 10 out of 13 tasks, assigns nonzero combination weights to fewer kernel functions, compared to MKL. This leads to a decrease in kernel calculations, and therefore test time for new instances. As an extreme case, on the Y8 task, RMKL converges to a point where all d_m coefficients except for the first kernel function are equal to 10 and are effectively pruned. For this task, we obtain combination rules that use only one kernel function (k_{Pfam}) with higher average accuracy. RMKL uses fewer kernels on Y2, Y3, Y4, Y7, Y10, and Y11 tasks by assigning a larger regularization parameter to k_{Exp}. This can be interpreted as an indication that gene expression profiles do not give useful information for these classification tasks and can safely be removed from the ensemble of kernels.

Table 5.9 lists the average accuracy and support vector percentages for the second subset of kernels. As on the first subset, the regularized variant achieves similar accuracy results and statistically significantly reduces the support vector percentage on six out of 13 tasks, increasing on four tasks. With direct comparison, the numbers of wins increase to six and eight for the average accuracy on the test set and support vectors stored, respectively. On this second subset, the Wilcoxon's signed-rank test does not report a difference between MKL and RMKL in terms of average accuracies on the test set nor support vector counts.

Table 5.9 also gives the combination weights ($\{\eta_m\}_{m=1}^P$) for both methods and the regularization parameters ($\{d_m\}_{m=1}^P$) found by RMKL. On 12 out of 13 tasks, RMKL assigns nonzero weights to fewer kernel functions. As mentioned before, this leads to a significant reduction in the total time needed to calculate the output for a given test input. Note that here, we do not explicitly penalize nonzero weights; the number of kernels used decreases as a part of the regularization process. On the Y8 task, RMKL converges to a point where regularization coefficient for k_{SW} is smaller than 1. We can say that the Smith-Waterman kernel provides the most informative similarity measure for this classification task and its combination weight goes up to 4.10, removing all other kernels. A similar behavior is also observed for k_{PfamE} on Y1, Y4, Y5, Y6, and Y12 tasks. On all these tasks, RMKL assigns combination weights greater than

Table 5.9. The average test accuracies, support vector percentages, combination weights, and regularization parameters with (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}) combination on the protein function prediction experiments.

	MKL		RMKL		
	Test Acc.	SV	Test Acc.	SV	
	η_1-η_2-η_3-η_4-η_5-η_6		η_1-η_2-η_3-η_4-η_5-η_6		d_1-d_2-d_3-d_4-d_5-d_6
Y1	80.72±0.74	93.24±0.78	80.68±0.52	**89.29±1.28**	
	0.17-0.03-0.10-0.16-0.05-0.48		0.45-0.17-0.20-0.03-0.10-0.00		1.00-1.02-1.06-1.18-1.02-2.52
Y2	94.32±0.21	75.74±2.90	94.48±0.22	**72.15±3.05**	
	0.00-0.01-0.12-0.21-0.06-0.59		0.00-0.00-0.21-0.00-0.07-0.71		1.00-2.06-1.00-2.06-1.00-1.00
Y3	87.51±0.38	81.13±1.69	87.53±0.34	**73.17±1.14**	
	0.04-0.01-0.15-0.30-0.05-0.45		0.23-0.04-0.22-0.40-0.11-0.00		1.00-1.00-1.00-1.00-1.00-2.08
Y4	85.63±0.48	86.10±1.11	85.53±0.77	**76.23±1.30**	
	0.15-0.03-0.20-0.23-0.06-0.33		0.45-0.04-0.38-0.01-0.04-0.00		1.00-1.12-1.05-1.37-1.25-1.50
Y5	93.68±0.50	62.69±3.23	94.59±0.23	**34.16±1.49**	
	0.14-0.01-0.16-0.09-0.19-0.40		0.50-0.01-0.01-0.00-1.01-0.00		1.00-1.21-1.32-1.46-0.68-1.48
Y6	87.44±0.74	93.01±0.96	87.90±0.61	**84.12±1.38**	
	0.11-0.01-0.11-0.19-0.04-0.54		0.42-0.14-0.33-0.00-0.11-0.00		1.00-1.00-1.00-2.06-1.00-2.06
Y7	91.05±0.24	**75.72±1.28**	90.70±0.35	81.55±1.09	
	0.08-0.01-0.28-0.23-0.02-0.38		0.35-0.00-0.49-0.11-0.00-0.00		1.00-1.57-1.00-1.22-1.46-1.50
Y8	93.78±0.43	**57.60±2.13**	93.77±0.31	81.87±1.66	
	0.01-0.03-0.18-0.24-0.03-0.52		0.00-0.00-0.00-0.00-0.00-4.10		1.00-1.34-1.36-1.17-1.29-0.49
Y9	95.00±0.32	76.14±2.69	94.99±0.31	72.54±3.77	
	0.05-0.27-0.20-0.11-0.04-0.33		0.04-0.00-0.29-0.00-0.08-0.58		1.00-2.06-1.00-2.06-1.00-1.00
Y10	89.56±0.36	**70.95±2.80**	89.58±0.36	89.83±1.49	
	0.00-0.00-0.27-0.22-0.02-0.48		0.00-0.00-0.27-0.23-0.00-0.50		1.00-1.00-1.00-1.00-2.08-1.00
Y11	94.43±0.11	87.63±1.79	94.43±0.11	87.63±1.79	
	0.00-0.03-0.19-0.24-0.02-0.53		0.00-0.03-0.19-0.24-0.02-0.53		1.00-1.00-1.00-1.00-1.00-1.00
Y12	96.41±0.32	**36.42±1.02**	96.22±0.29	46.70±1.53	
	0.43-0.02-0.19-0.04-0.01-0.31		0.68-0.00-0.15-0.03-0.00-0.02		1.00-1.38-1.25-1.25-1.38-1.25
Y13	97.80±0.09	61.70±9.56	97.83±0.11	50.16±9.26	
	0.00-0.18-0.12-0.15-0.07-0.50		0.00-0.00-0.18-0.00-0.15-0.67		1.00-2.06-1.00-2.06-1.00-1.00
5×2 cv Paired F Test (W-T-L)			0-13-0	6-3-4	
Direct Comparison (W-T-L)			6-1-6	8-1-4	
Wilcoxon's Rank Test (W/T/L)			T	T	

0.42 to k_{PfamE}. The reduction in the number of kernel functions used in the decision function after training is also observed in this set of experiments.

<u>5.2.2.3. Comparison with Other Methods.</u> Figure 5.2 compares MKL with RMKL, EMKL, and RWKL methods using (k_L-k_P-k_G) combination on the bioinformatics data sets. We see that RMKL obtains better or similar accuracy results, compared to MKL on these data sets except POLYADENYLATION. RMKL, EMKL, and RWKL obtain similar results for all data sets. However, RMKL always uses fewer or as many kernels on all tasks. On the ARABIDOPSIS data set for example, as shown in Table 5.4, RMKL favors the Gaussian kernel instead of the linear and the second-degree polynomial kernels. On this data set, if we use the eigenvalues to decide on the regularization parameters, we penalize the Gaussian kernel due to the high number of large eigenvalues and fail to obtain the result obtained by RMKL.

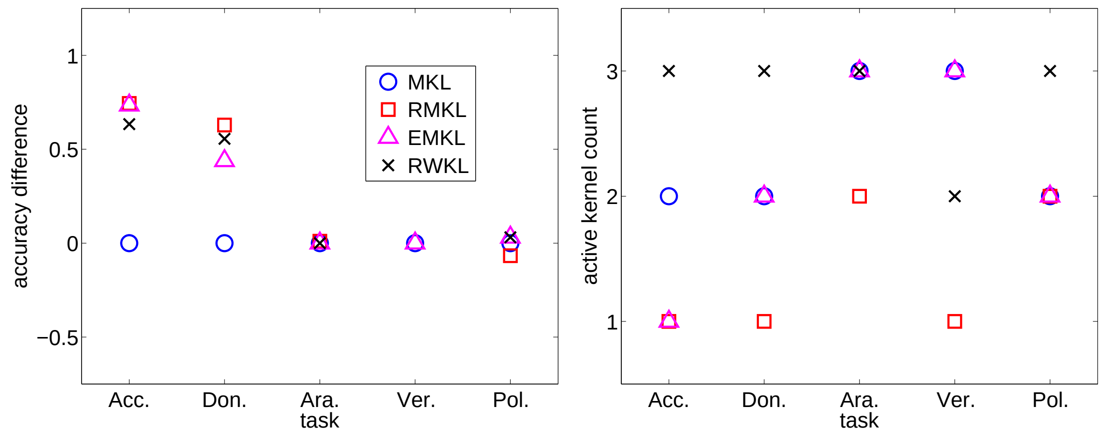


Figure 5.2. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with (k_L-k_P-k_G) combination. In accuracy comparisons, the average accuracy of MKL is used as the baseline performance.

Figure 5.3 compares MKL, RMKL, EMKL, and RWKL methods with (k_{FAC}-k_{FOU}-k_{KAR}-k_{MOR}-k_{PIX}-k_{ZER}) combination on the MULTIFEAT data set. We see that RMKL obtains statistically similar accuracy results, compared to other methods on these two tasks, by using fewer or as many kernels.

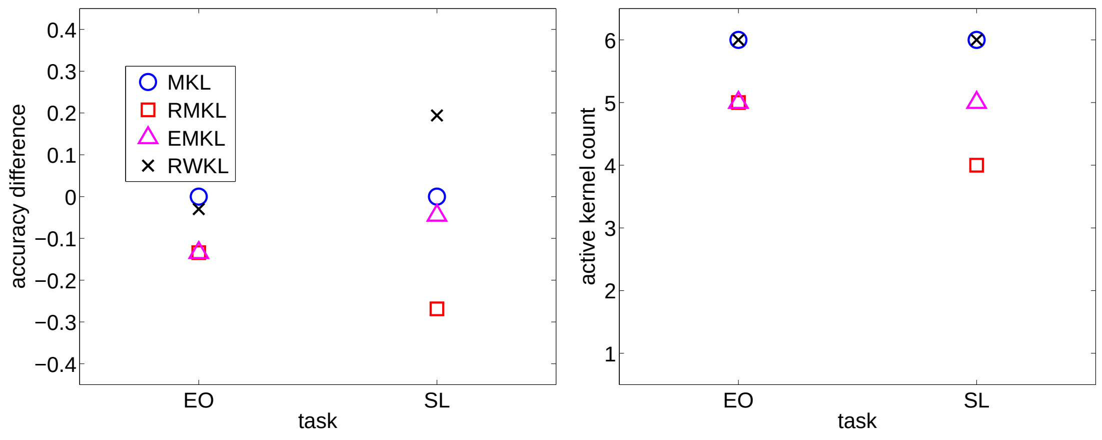


Figure 5.3. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with (k_{FAC}-k_{FOU}-k_{KAR}-k_{MOR}-k_{PIX}-k_{ZER}) combination.

When we compare MKL, RMKL, EMKL, and RWKL on the protein function prediction problem with (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) combination, we see in Figure 5.4 that RMKL, EMKL, and RWKL generally obtain better accuracy results than MKL. However, there are significant differences between the number of used kernels by RMKL and other methods on almost all tasks. RMKL obtains smaller kernel ensembles due to the regularization and sparsity effects of MKL and eliminates the redundant kernels/data sources.

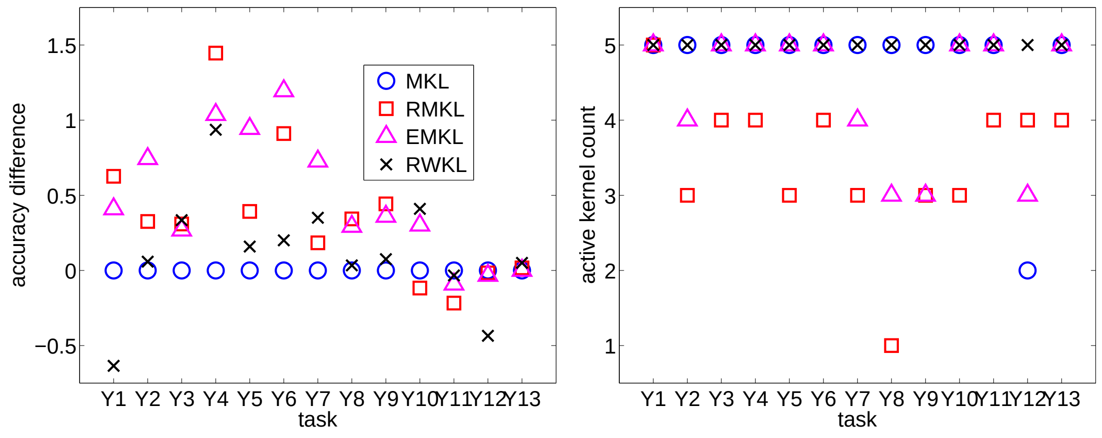


Figure 5.4. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) combination.

When we make the same comparison with (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}) combination, Figure 5.5 shows that all methods obtains comparable accuracy results on almost all tasks. RMKL again uses fewer kernels compared to the other three methods without diminishing accuracy.

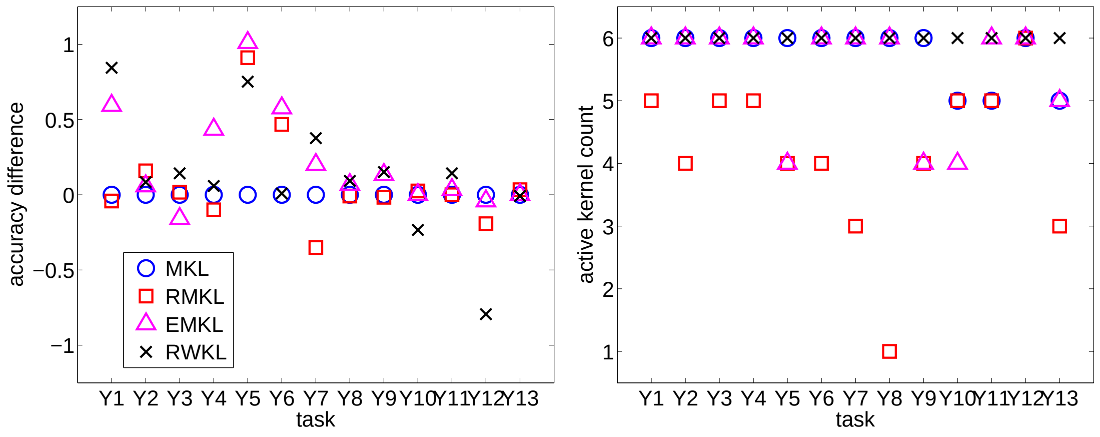
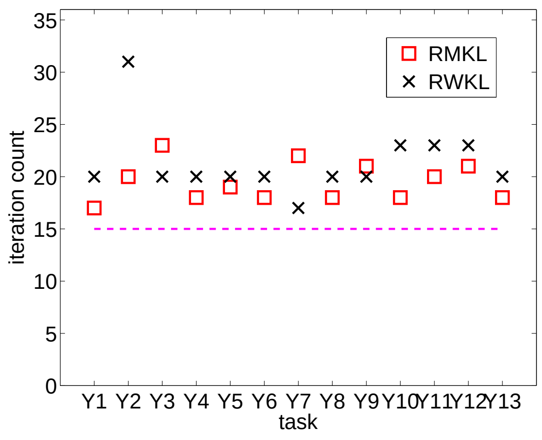
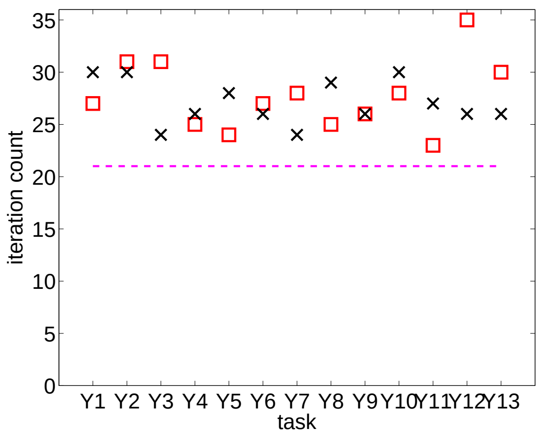


Figure 5.5. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}) combination.

The running time of our proposed method is directly related to the running time of the MKL solver and the convergence speed of RSM. The convergence time can be long especially with large number of kernels, (in practice, we see convergence in terms of iterations) but it will always be faster and more detailed than exhaustive grid search. The running time can be decreased by using a subset of training points in the intermediate steps of RSM; once the best $\{d_m\}_{m=1}^{P}$ set is found, the whole training set can be used for final training before test. Figure 5.6 shows the number of iterations performed by RMKL and RWKL for all tasks of protein function prediction problem. The running time of RMKL and EMKL is directly related to the number of times the MKL solver is called. RMKL calls the MKL solver 15 to 35 times for all tasks of protein function prediction problem, whereas EMKL calls the solver 11 times for the different γ values $(0, 0.1, 0.2, \ldots, 1.0)$ we try. However, RWKL calls a canonical SVM solver at each iteration and 15 to 30 times in total. Clearly, the running time of RWKL is smaller than both RMKL and EMKL due to the complexity difference between the optimization problems of SVM and MKL.



(a) $P = 5$



(b) $P = 6$

Figure 5.6. The number of iterations performed by RMKL and RWKL with (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) and (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}) combinations. Dashed lines show the number of iterations required to initialize RSM.

RSM converges in 1 to 14 additional iterations for RMKL after the initialization phase and requires much fewer iterations than grid search, whereas grid search requires 81 (3^4) and 243 (3^5) iterations for the cases of $P = 5$ and $P = 6$, respectively, if we use three levels for $\{d_m\}_{m=1}^{P}$ parameters and arbitrarily fix one of them. Fitting a quadratic approximation in RSM requires $\mathcal{O}(P^2)$ sample points, which may be costly when P, the number of kernels, is large. One can fit a first-order RSM model using $\mathcal{O}(P)$ sample points to initialize and use gradient-descent to find the next point to be sampled.

5.2.3. Cost-Conscious Multiple Kernel Learning Experiments

In this section, we perform experiments using the cost-conscious MKL approach of Section 2.7. In this set of experiments, C parameter is selected from $\{0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000\}$ using cross-validation. In the result tables, we report the average test accuracies, support vector percentages, total costs, and normalized combination weights. Total cost calculation is performed with the cost coefficients of cost-conscious variant to get comparable results. The average test accuracies, support vector percentages, and total costs are made bold if the difference between MKL and the cost-conscious variant is statistically significant using the 5×2 cv paired F

test (Alpaydın, 1999). We also report the count of (W)ins-(T)ies-(L)osses of kernel combination with the cost-conscious MKL from direct comparison and the 5×2 cv paired F test. The Wilcoxon's signed-rank test is used to compare the two variants over a number of data sets in terms of average accuracie, support vector percentages, and total costs (Wilcoxon, 1945). The result is shown as (W)in, (T)ie, or (L)oss. For both statistical tests, the significance level, α, is taken as 0.05.

5.2.3.1. Kernel Selection on UCI Data Sets. We perform the experiments on 12 benchmark data sets (Banana, Heart, Ionosphere, Liverdisorder, Optdigits, Pendigits, Pima, Ringnorm, Sonar, Spambase, Twonorm, and Wdbc) from the UCI Machine Learning Repository. Optdigits and Pendigits data sets are optical and pen-based digit recognition problems, respectively. Two-class subsets of these data sets (1vs8 and 3vs9 for Optdigits, 0vs8, 1vs7, and 5vs9 for Pendigits) are taken to obtain binary classification problems.

Three different kernel functions are used in this part: the linear kernel, the polynomial kernel, and the Gaussian kernel. We use the second-degree ($q = 2$) polynomial kernel and estimate s in the Gaussian kernel as in (2.10). Outputs obtained from each kernel function are in different scales. In order to avoid this scale problem, kernel outputs are normalized by using (2.9). We experiment two scenarios on (k_L-k_P-k_G): In the first one, all kernels have equal cost ($d_L = 1.00$, $d_P = 1.00$, $d_G = 1.00$) and in the second, their cost increase as we go from linear to polynomial to Gaussian ($d_L = 1.00$, $d_P = \sqrt{2} = 1.41$, $d_G = 2.00$). The polynomial kernel is slightly more costly than the linear kernel because of taking the second power and the Gaussian kernel is more complicated than that because of the $\exp(\cdot)$ function. These values we use are rough estimates; exact values depend on the particular hardware implementation.

On all 12 data sets, the comparison of results by MKL and cost-conscious MKL is given in Table 5.10. We can see that Pendigits (5vs9), Pima, and Wdbc use only the linear kernel when we increase d_P and d_G and achieve statistically similar accuracy results. Removing the polynomial and the Gaussian kernel from the ensemble by

Table 5.10. The average test accuracies, support vector percentages, total costs and normalized combination weights with (k_L-k_P-k_G) combination on benchmark data sets. The first line is the case where all kernels have equal cost and the second line is where complex kernels are penalized.

Data Set	Test Acc.	SV	Total Cost	η_L-η_P-η_G
Banana	83.17±0.52	90.68±0.11	70.14±0.08	0.00-0.03-0.97
	83.27±1.05	83.27±0.11	64.40±0.08	0.00-0.04-0.96
Heart	79.67±1.74	79.44±3.28	18.00±0.74	1.00-0.00-0.00
	79.67±1.74	79.44±3.28	18.00±0.74	1.00-0.00-0.00
Ionosphere	93.85±1.13	64.36±2.73	54.25±7.54	0.01-0.46-0.53
	90.94±1.57	**46.67±2.52**	44.65±7.54	0.29-0.60-0.10
Liverdisorder	64.17±4.14	93.30±1.54	57.01±20.32	0.02-0.00-0.98
	65.91±3.17	**80.87±3.60**	66.57±20.32	0.67-0.30-0.03
Optdigits (1vs8)	98.01±0.14	28.21±2.68	28.21±1.00	0.27-0.38-0.34
	98.01±0.14	**19.12±1.83**	**10.46±1.00**	0.53-0.47-0.00
Optdigits (3vs9)	96.71±1.53	25.46±4.19	25.46±1.15	0.44-0.20-0.36
	97.37±1.39	**15.25±2.10**	**8.34±1.15**	0.73-0.27-0.00
Pendigits (0vs8)	99.30±0.11	11.70±0.53	11.70±0.11	0.18-0.71-0.11
	99.80±0.00	**9.10±0.11**	**9.10±0.11**	0.25-0.66-0.09
Pendigits (1vs7)	100.00±0.00	14.00±1.69	14.00±1.58	0.39-0.24-0.37
	100.00±0.00	**6.50±1.58**	**6.50±1.58**	0.68-0.23-0.09
Pendigits (5vs9)	99.20±0.21	33.20±0.21	18.16±0.05	0.79-0.21-0.00
	99.20±0.21	32.60±0.21	**7.39±0.05**	1.00-0.00-0.00
Pima	73.09±0.75	68.91±1.62	20.14±0.37	0.97-0.03-0.00
	72.93±1.01	68.79±1.65	15.58±0.37	1.00-0.00-0.00
Ringnorm	**98.01±0.63**	**26.85±0.00**	**12.17±0.79**	0.00-0.00-1.00
	87.25±0.42	82.46±1.16	56.05±0.79	0.75-0.00-0.25
Sonar	81.14±3.14	86.96±2.46	86.96±3.10	0.14-0.33-0.54
	80.29±3.61	**69.13±5.68**	**37.81±3.10**	0.48-0.52-0.00
Spambase	92.53±0.10	42.99±4.33	42.99±6.23	0.39-0.18-0.43
	92.43±0.00	**31.06±2.53**	**12.40±6.23**	0.96-0.04-0.00
Twonorm	97.20±0.00	89.10±0.11	20.18±0.05	1.00-0.00-0.00
	97.20±0.00	89.00±0.21	20.16±0.05	1.00-0.00-0.00
Wdbc	95.45±0.91	22.26±3.21	16.97±0.36	0.70-0.06-0.24
	94.97±0.97	17.00±1.59	**3.85±0.36**	1.00-0.00-0.00
(W-T-L)	2-12-1	8-6-1	8-6-1	5×2 cv Paired F Test
(W-T-L)	4-5-6	13-1-1	12-1-2	Direct Comparison
(W/T/L)	T	W	W	Wilcoxon's Rank Test

penalizing them decreases total cost significantly for Pendigits (5vs9) and Wdbc. The cost-conscious MKL assigns zero weight to the Gaussian kernel for Optdigits (1vs8 and 3vs9), Sonar, and Spambase data sets, whereas the equal cost variant uses the Gaussian kernel with weights larger than 0.30. Nonlinear data sets such as Banana and Ringnorm continue using the Gaussian kernel even if its cost is increased to 2. However the average test accuracy in Ringnorm data set is reduced drastically after changing the combination weight of the Gaussian kernel from 1.00 to 0.25. By comparing two variants over all data sets, we see that the cost-conscious MKL achieves similar accuracy results according to both the 5×2 cv paired F test (12 ties out of 15 tasks) and the Wilcoxon's signed rank test. The 5×2 cv paired F test reports that total cost is decreased significantly over eight out of 15 tasks. Reduction in total cost is also reported to be statistically significant using the Wilcoxon's signed rank test over 15 tasks.

5.2.3.2. Representation Selection on Handwritten Digit Recognition Data Set. Kernel combination can also be used to combine different data representations or modalities. In this case, it can be the case that extracting different representations or modalities may have costs associated with them and we do not want to use a costly one unless it is deemed necessary for classification. For example, Pendigits data set has four different representations: Dyn, Sta4, Sta8, and Sta16 (Alimoğlu and Alpaydın, 1997). Dyn contains eight successive pen points on two-dimensional coordinate system and is used when combining kernel functions on Pendigits data set. Sta16 is 16×16 image bitmap representation of the corresponding training instance formed by connecting the points in the Dyn representation by line segments. Sta4 and Sta8 are 4×4 and 8×8 subsampled bitmap representations of Sta16, respectively. Figure 5.7 illustrates Dyn, Sta16, Sta8, and Sta4 representations on a sample data instance.

In our experiments, we use linear kernels over four different representations of Pendigits data set. We form two sets of experiments as follows: (a) (k_{Dyn}-k_{Sta16}) with (d_{Dyn}-d_{Sta16}) taken as (1.00-1.00) and (1.00-4.00), increasing the cost of forming the image representation, and, (b) (k_{Dyn}-k_{Sta4}-k_{Sta8}-k_{Sta16}) with (d_{Dyn}-d_{Sta4}-d_{Sta8}-

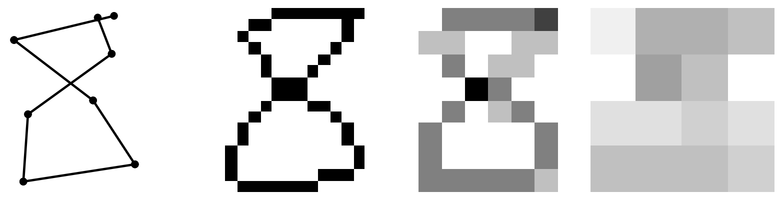

(a) Dyn (b) Sta16 (c) Sta8 (d) Sta4

Figure 5.7. Four different representations for an example digit eight.

d_{Sta16}) taken as (1.00-1.00-1.00-1.00) and (1.00-1.00-2.00-4.00), increasing the cost of representation with higher dimensionalities.

As we see in Table 5.11, when both representations, Dyn and Sta16, have equal cost, both are chosen for 0vs8 and 1vs7. When we increase the cost of the Sta16 to 4, only Dyn representation is used for 1vs7. The average accuracy percentage for all three tasks remains the same according to the 5×2 cv paired F test results with increasing d_{Sta16}. Total cost in testing phase decreases because of two factors: (a) The average support vector percentages are decreased. (b) k_{Sta16} is not evaluated when η_{Sta16} is equal to 0.

Table 5.11. The average test accuracies, support vector percentages, total costs and normalized combination weights with (Dyn-Sta16) combination.

Data Set	Test Acc.	SV	Total Cost	η_{Dyn}-η_{Sta16}
Pendigits (0vs8)	99.40±0.21	15.10±0.95	15.10±0.21	0.21-0.79
	99.00±0.42	**8.80±0.21**	**8.80±0.21**	0.75-0.25
Pendigits (1vs7)	99.70±0.32	11.00±0.21	11.00±0.02	0.35-0.65
	100.00±0.00	**3.70±0.11**	**0.74±0.02**	1.00-0.00
Pendigits (5vs9)	99.20±0.21	32.80±0.21	6.56±0.08	1.00-0.00
	99.20±0.21	32.80±0.42	6.56±0.08	1.00-0.00

When we combine all four representations and penalize kernels proportional to their dimensionality (see Table 5.12), we obtain accuracy results that are statistically the same, but using only two representations (Dyn and Sta4) for 5vs9 and three representations (Dyn, Sta4, and Sta8) for 0vs8 and 1vs7. The cost-conscious MKL also uses significantly fewer support vectors in all cases. Estimated total cost for both cases decrease drastically as a result of using fewer kernels and support vectors.

Table 5.12. The average test accuracies, support vector percentages, total costs and normalized combination weights with (Dyn-Sta4-Sta8-Sta16) combination.

Data Set	Test Acc.	SV	Total Cost	η_{Dyn}-η_{Sta4}-η_{Sta8}-η_{Sta16}
Pendigits (0vs8)	99.50±0.11	10.70±0.11	10.70±0.33	0.24-0.16-0.33-0.26
	99.30±0.32	**5.80±0.42**	**2.90±0.21**	0.48-0.38-0.14-0.00
Pendigits (1vs7)	99.90±0.11	9.70±0.53	9.70±1.33	0.32-0.31-0.31-0.06
	99.70±0.32	**3.00±0.42**	**1.50±0.21**	0.45-0.51-0.04-0.00
Pendigits (5vs9)	99.30±0.11	12.90±0.95	12.90±0.04	0.35-0.12-0.10-0.44
	99.50±0.11	**10.30±0.11**	**2.58±0.03**	0.64-0.36-0.00-0.00

5.2.3.3. Kernel/Representation Selection on Bioinformatics Data Sets.

We perform protein location prediction and protein function prediction experiments on the MIPS CYGD (Mewes *et al.*, 2000). CYGD assigns subcellular locations for 2318 and 1150 proteins according to whether they participate in the membrane and the ribosome, respectively. We combine the seven kernel functions used also in Lanckriet *et al.* (2004b) for comparing equal cost and cost-conscious variants of MKL.

k_{SW} and k_B are generated from protein sequences using Smith-Waterman (SW) and the BLAST pairwise sequence comparison algorithms, respectively. k_{Pfam} are also generated from protein sequences by replacing pairwise comparison scores with the expectation values obtained from hidden Markov models in the Pfam database. k_{FFT} is calculated by comparing the frequency content of the hydropathy profiles of the two proteins. k_{LI} is the inner product between interaction values for a pair of proteins. k_D is the diffusion kernel calculated over the graph constructed by using the same interaction data used in k_{LI}. k_E is the Gaussian kernel calculated over microarray gene expression measurements.

k_{SW}, k_B, and k_{Pfam} require pairwise comparison scores for protein sequences, so, they are computationally expensive. The same concern is also valid for k_{FFT}. k_{LI}

obtained by inner product over protein interactions is a simple kernel. k_D requires constructing a graph from protein interactions and calculating a similarity measure based on a random walk on this graph. k_E is also a cheap kernel, which simply evaluates the Gaussian kernel function over 441 dimensional gene expressions.

Two different cost combinations are formed by considering all seven kernels despite the fact that k_E and k_{FFT} are not much relevant for membrane and ribosomal protein recognition tasks, respectively. We consider the following two (d_{SW}-d_B-d_{Pfam}-d_{FFT}-d_{LI}-d_D-d_E) combinations: (a) (1.00-1.00-1.00-1.00-1.00-1.00-1.00), all kernels are considered with equal cost coefficients as a base case, and, (b) (1.41-1.41-1.41-1.41-1.00-2.00-1.00), the diffusion kernel (k_D) is assigned the largest cost coefficient due to its computational complexity, k_{LI} and k_E are given cost coefficients smaller than those of sequence based kernels (k_{SW}-k_B-k_{Pfam}-k_{FFT}) due to their simplicity. The cost values we assign here are used as rough estimates to give us an ordering of the costs. Exact values depend on the implementation of these kernel functions and the time/space complexity of the data structures and the algorithms that are used.

Table 5.13 summarizes the results for ribosomal and membrane protein recognition problem. Cost-conscious variant uses statistically significantly fewer support vectors compared to the equal cost variant in membrane protein recognition. As an important result, it can be observed that discarding k_D (i.e., $d_D = 2$), which is computationally expensive, and using protein sequence based kernels with a larger cost than k_{LI} achieves similar classification results by improving total cost for both tasks.

Table 5.13. The average test accuracies, support vector percentages, total costs and normalized combination weights for protein recognition tasks.

Task	Test Acc.	SV	Total Cost	η_{SW}-η_B-η_{Pfam}-η_{FFT}-η_{LI}-η_D-η_E
Membrane	86.30±0.61	84.42±1.03	75.34±5.09	0.28-0.31-0.11-0.05-0.00-0.15-0.10
	85.40±0.55	**71.33±1.67**	**37.81±5.09**	0.00-0.26-0.06-0.00-0.07-0.00-0.60
Ribosomal	98.99±0.26	18.56±1.56	6.86±1.96	0.01-0.00-0.00-0.16-0.03-0.00-0.80
	99.07±0.18	18.61±1.46	6.08±1.96	0.00-0.00-0.00-0.01-0.03-0.00-0.96

We also perform experiments on protein function prediction data set desribed in Subsection 5.2.2. k_{Pfam} is the inner product between binary representations of protein sequences to Pfam domains. k_{PfamE} is an enriched variant of k_{Pfam} obtained by using additional domains. k_{TAP}, k_{Phys}, and k_{Gen} are three different diffusion kernels calculated over the graphs constructed from three different types of protein interactions: co-participation in a protein complex, genetic interactions, and protein-protein interactions, respectively. k_{Exp} and k_{ExpG} are calculated from cell cycle gene expression measurements. k_{Exp} is a binary kernel function that is determined by using Pearson correlation of a pair of expression profiles. k_{ExpG} is the Gaussian kernel defined on the expression profiles. k_{SW} is obtained by applying SW algorithm on protein sequences.

Two different kernel subsets described in Lanckriet *et al.* (2004a) are also used for experiments: (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) and (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}). For the first subset, k_{Pfam} and diffusion kernels are assigned cost coefficients higher ($d_{Pfam} = 1.41$ and $d_{TAP} = d_{Phys} = d_{Gen} = 2.00$) than k_{Exp} ($d_{Exp} = 1.00$). Likewise, k_{PfamE}, k_{SW}, and diffusion kernels are penalized ($d_{PfamE} = d_{SW} = 1.41$ and $d_{TAP} = d_{Phys} = d_{Gen} = 2.00$) in the second subset.

The results for each binary classification task in the first subset are given in Table 5.14. We see that avoiding the expensive kernels such as diffusion kernels leads to significant accuracy loss and increase in the number of support vectors stored. The cost-conscious MKL uses k_{Exp} only in four out of 13 tasks. In other tasks except Y12, k_{Exp} is used with a combination weight between 0.82 and 1. Support vector percentages are increased in nearly all cases but total kernel evaluation will be decreased for test instances due to the unused kernels. This result can also be seen from the estimated total cost results, which has an obvious decreasing trend for the cost-conscious variant. The 5×2 cv paired F test and the Wilcoxon's signed rank test report significant win for total cost and significant loss for the average test accuracy and support vector percentages, indicating that in this case, the expensive kernels are worth their costs.

Table 5.15 lists the results for the second subset. Cost-conscious variant uses k_{ExpG} heavily with combination weights ranging from 0.39 to 0.89. Equal cost variant

Table 5.14. The average test accuracies, support vector percentages, total costs and normalized combination weights with (k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}) combination.

Task	Test Acc.	SV	Total Cost	η_{Pfam}-η_{TAP}-η_{Phys}-η_{Gen}-η_{Exp}
Y1	**78.20**±**0.84**	**91.08**±**0.85**	91.08±26.82	0.67-0.03-0.07-0.17-0.06
	72.70±0.56	97.03±0.59	50.96±26.82	0.14-0.00-0.00-0.01-0.85
Y2	93.29±0.10	**91.10**±**2.33**	88.92±0.08	0.23-0.01-0.14-0.18-0.44
	93.24±0.06	96.54±0.69	**11.47**±**0.08**	0.00-0.00-0.00-0.00-1.00
Y3	**86.40**±**0.46**	95.68±0.86	95.68±14.66	0.11-0.01-0.18-0.32-0.37
	83.96±0.35	94.62±1.43	**56.27**±**14.66**	0.16-0.00-0.00-0.01-0.83
Y4	**84.81**±**0.56**	93.17±1.07	93.17±18.68	0.22-0.03-0.23-0.21-0.31
	82.07±0.49	94.48±1.18	60.71±18.68	0.16-0.01-0.01-0.00-0.82
Y5	91.51±0.17	**75.42**±**1.54**	75.42±23.51	0.19-0.01-0.13-0.25-0.41
	91.50±0.19	90.96±1.77	60.76±23.51	0.15-0.00-0.01-0.01-0.84
Y6	**86.75**±**0.45**	91.74±1.25	76.48±29.43	0.63-0.00-0.13-0.23-0.01
	83.84±0.66	95.46±1.20	54.40±29.43	0.16-0.00-0.00-0.01-0.82
Y7	**90.20**±**0.41**	**83.09**±**1.92**	75.09±0.18	0.18-0.03-0.28-0.23-0.27
	87.01±0.29	95.46±0.63	**27.39**±**0.18**	0.12-0.00-0.00-0.00-0.88
Y8	92.93±0.16	91.67±3.51	82.92±0.23	0.34-0.01-0.10-0.14-0.42
	92.73±0.16	95.91±0.81	**27.52**±**0.23**	0.10-0.00-0.00-0.00-0.90
Y9	94.59±0.14	**84.71**±**2.71**	84.71±0.11	0.16-0.11-0.22-0.11-0.39
	94.57±0.00	96.55±0.94	**11.47**±**0.11**	0.00-0.00-0.00-0.00-1.00
Y10	**89.97**±**0.23**	94.88±0.53	79.08±17.56	0.09-0.00-0.24-0.23-0.44
	88.54±0.48	92.54±1.85	**44.18**±**17.56**	0.15-0.00-0.02-0.00-0.83
Y11	94.66±0.11	92.97±1.92	75.30±0.04	0.07-0.00-0.17-0.25-0.51
	94.65±0.00	97.22±0.38	**11.55**±**0.04**	0.00-0.00-0.00-0.00-1.00
Y12	94.87±0.35	75.80±2.17	32.56±0.87	0.99-0.01-0.00-0.00-0.00
	94.10±0.62	83.02±3.02	23.82±0.87	0.59-0.00-0.00-0.00-0.41
Y13	97.74±0.00	86.64±8.66	85.43±0.06	0.12-0.05-0.10-0.07-0.66
	97.74±0.00	96.79±0.54	**11.50**±**0.06**	0.00-0.00-0.00-0.00-1.00
(W-T-L)	0-7-6	0-8-5	8-5-0	5×2 cv Paired *F* Test
(W-T-L)	0-1-12	2-0-11	13-0-0	Direct Comparison
(W/T/L)	L	L	W	Wilcoxon's Signed Rank Test

Table 5.15. The average test accuracies, support vector percentages, total costs and normalized combination weights with (k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}) combination.

Task	Test Acc.	SV	Total Cost	η_{PfamE}-η_{TAP}-η_{Phys}-η_{Gen}-η_{ExpG}-η_{SW}
Y1	79.15±0.77	92.74±0.84	92.74±8.57	0.16-0.02-0.09-0.16-0.05-0.51
	78.15±0.49	**86.26±1.20**	**38.88±8.57**	0.11-0.00-0.00-0.00-0.56-0.33
Y2	94.13±0.29	70.44±4.10	66.03±3.99	0.01-0.01-0.16-0.21-0.09-0.52
	93.94±0.49	**42.96±3.49**	**13.78±3.99**	0.00-0.00-0.00-0.00-0.63-0.36
Y3	86.40±0.54	90.47±1.07	86.85±5.76	0.05-0.01-0.16-0.29-0.05-0.44
	85.03±0.50	**66.71±3.20**	**36.29±5.76**	0.14-0.00-0.00-0.05-0.75-0.06
Y4	**85.64±0.58**	84.27±1.67	84.27±0.37	0.17-0.03-0.20-0.18-0.06-0.37
	82.32±0.76	**62.67±0.82**	**28.15±0.37**	0.34-0.01-0.00-0.00-0.65-0.00
Y5	93.12±0.50	60.38±2.86	60.38±4.73	0.17-0.01-0.12-0.15-0.22-0.32
	94.54±0.42	**49.11±2.85**	**27.18±4.73**	0.10-0.00-0.00-0.00-0.64-0.26
Y6	86.89±0.37	84.23±0.94	82.53±0.62	0.14-0.01-0.12-0.24-0.04-0.47
	85.96±0.85	**77.61±1.59**	**30.23±0.62**	0.10-0.00-0.00-0.00-0.50-0.40
Y7	**91.14±0.46**	89.77±1.19	87.96±0.87	0.09-0.04-0.28-0.24-0.03-0.32
	88.07±0.54	**75.24±2.25**	**29.31±0.87**	0.24-0.00-0.00-0.00-0.39-0.37
Y8	93.66±0.19	83.07±3.94	70.03±3.43	0.00-0.04-0.18-0.22-0.03-0.52
	93.41±0.31	**53.28±1.69**	**15.33±3.43**	0.00-0.00-0.00-0.00-0.65-0.34
Y9	94.67±0.35	40.18±5.08	39.40±6.79	0.02-0.14-0.24-0.12-0.01-0.47
	94.72±0.38	57.95±3.32	**23.52±6.79**	0.02-0.00-0.00-0.00-0.60-0.37
Y10	**90.09±0.25**	70.98±1.62	49.26±6.70	0.00-0.00-0.23-0.19-0.02-0.57
	88.65±0.09	71.62±3.64	**25.36±6.70**	0.00-0.00-0.01-0.00-0.61-0.38
Y11	94.65±0.00	86.89±2.20	58.53±2.91	0.00-0.01-0.16-0.13-0.01-0.68
	94.65±0.00	**58.95±3.44**	**15.34±2.91**	0.00-0.00-0.00-0.00-0.70-0.30
Y12	96.01±0.24	54.92±3.24	45.70±2.61	0.39-0.11-0.07-0.02-0.02-0.37
	95.58±0.59	**38.89±1.58**	**15.94±2.61**	0.34-0.00-0.00-0.00-0.46-0.20
Y13	97.68±0.10	63.12±5.14	54.04±2.33	0.00-0.24-0.10-0.11-0.03-0.51
	97.74±0.00	**30.47±2.17**	**8.82±2.33**	0.02-0.00-0.00-0.00-0.89-0.09
(W-T-L)	0-10-3	11-2-0	13-0-0	5×2 cv Paired F Test
(W-T-L)	3-1-9	11-0-2	13-0-0	Direct Comparison
(W/T/L)	L	W	W	Wilcoxon's Signed Rank Test

uses k_{SW} with a significant weight in all tasks. However cost-conscious variant prefers to use k_{SW} with a smaller coefficient. Different from the first subset, support vector percentages also decrease significantly in 11 out of 13 tasks, in addition to a decrease in total cost. The 5×2 cv paired F test reports a significant loss for the average test accuracy in three tasks and the difference between the average test accuracies of two variants is significant according to the Wilcoxon's signed rank test. The Wilcoxon's signed rank test finds significant wins for the number of support vectors and cost.

5.3. Localized Multiple Kernel Learning Experiments

In this section, we report empirical performance of our method LMKL of Chapter 3 for classification and regression problems on several data sets and compare LMKL with SVM, SVR, and MKL (using the linear formulation of Bach *et al.* (2004)). In this set of experiments, C parameter is selected from $\{0.01, 0.1, 1, 10, 100\}$ using cross-validation. We use the 5×2 cv paired F test for statistical comparison (Alpaydın, 1999). The significance level, α, for the 5×2 cv paired F test is taken as 0.05.

5.3.1. Classification Experiments

We perform experiments on benchmark and image recognition classification data sets in order to compare with LMKL with other methods.

5.3.1.1. Combining Multiple Representations on Benchmark Data Sets. We compare SVM, MKL, and LMKL in terms of classification performance and model complexity (i.e., stored support vector percentage). We train SVMs with linear kernels calculated on each feature representation separately. We also train an SVM with a linear kernel calculated on the concatenation of all feature representations, which is referred to as All. MKL and LMKL combine linear kernels calculated on each feature representation. LMKL uses a single feature representation or the concatenation of all feature representations in the gating model. We use both softmax and sigmoid gating models in our experiments.

We perform experiments on the MultiFeat-SL data set desribed in Subsection 5.2.2. We use the concatenation of all feature representations in the gating model for this data set.

Table 5.16 lists the classification results on the MultiFeat-SL data set obtained by SVM, MKL, and LMKL. We see that SVM (All) is significantly more accurate than the best SVM with single feature representation, namely SVM (Fac), but with a significant increase in the number of support vectors. MKL is as accurate as SVM (All) but stores significantly more support vectors. LMKL with softmax gating is as accurate as SVM (All) using significantly fewer support vectors. LMKL with sigmoid gating is significantly more accurate than MKL, SVM (All), and single-kernel SVMs. It stores significantly fewer support vectors than MKL and SVM (All), and ties with SVM (Fac). For the MultiFeat-SL data set, the average kernel weights and the average number of active kernels (whose gating values are nonzero) calculated on the test set are given in Table 5.17. We see that both LMKL with softmax gating and LMKL with sigmoid gating use fewer kernels than MKL in the decision function. MKL uses all kernels with the same weight for all inputs; LMKL uses a different smaller subset for each input. By storing significantly fewer support vectors and using fewer active kernels, LMKL is significantly faster than MKL in the testing phase.

Instead of combining different feature representations, we can combine multiple copies of the same feature representation with LMKL. We combine multiple copies of linear kernels on the single best Fac representation using the sigmoid gating model on the same representation (see Figure 5.8). Even if we increase accuracy (not significantly) by increasing the number of copies of the kernels compared to SVM (Fac), we could not achieve the performance obtained by combining different representations with sigmoid gating. For example, LMKL with sigmoid gating and kernels over six different feature representations is better than LMKL with sigmoid gating and six copies of the kernel over the Fac representation in terms of both classification accuracy (though not significantly) and the number of support vectors stored (significantly) (see Table 5.16).

Table 5.16. Classification results on the MultiFeat-SL data set.

Method	Test Accuracy	Support Vector
SVM (Fac)	94.97±0.87	17.93±0.91
SVM (Fou)	90.54±1.11	28.90±1.69
SVM (Kar)	88.13±0.73	33.62±1.31
SVM (Mor)	69.61±0.14	61.90±0.49
SVM (Pix)	89.42±0.65	46.35±1.64
SVM (Zer)	89.12±0.63	26.27±1.67
SVM (All)	97.69±0.44	23.36±1.15
MKL	97.40±0.37	32.59±0.82
LMKL (softmax)	97.69±0.44	15.06±1.03
LMKL (sigmoid)	98.58±0.41	15.27±0.92
LMKL (6 Fac and sigmoid)	97.03±0.67	18.52±2.47

Table 5.17. Average kernel weights and number of active kernels on the MultiFeat-SL data set.

Method	Fac	Fou	Kar	Mor	Pix	Zer	Number of Active Kernels
MKL	0.2466	0.2968	0.0886	0.1464	0.1494	0.0722	6.00
LMKL (softmax)	0.1908	0.1333	0.1492	0.3335	0.1134	0.0797	2.43
LMKL (sigmoid)	0.5115	0.5192	0.5429	0.5566	0.5274	0.5132	5.36

We also perform experiments on the Internet Advertisements (Advert) data set from the UCI Machine Learning Repository, composed of five different feature representations (different bags of words) with some additional geometry information of the images, which is ignored in our experiments due to missing values. The properties of these feature representations are summarized in Table 5.18. The classification task is to predict whether an image is an advertisement or not. We use the Caption representation in the gating model due to its lower dimensionality compared to the other representations.

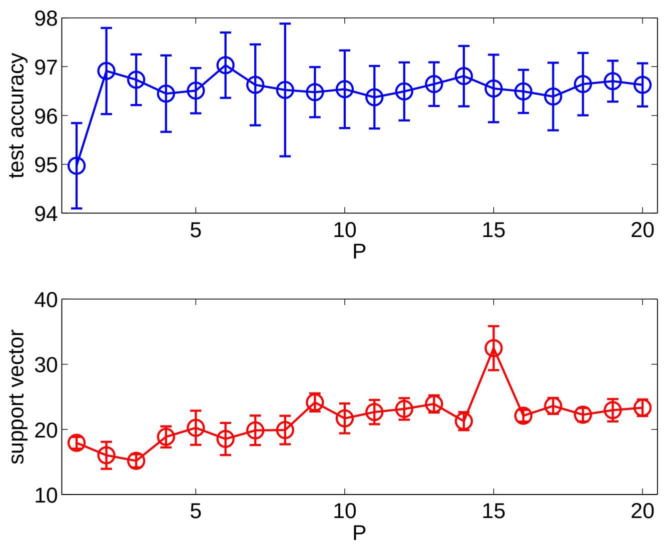


Figure 5.8. The average test accuracies and support vector percentages on the MultiFeat-SL data set obtained by LMKL with multiple copies of linear kernels and sigmoid gating on the Fac representation.

Table 5.18. Multiple feature representations in the Advert data set.

Name	Dimension	Data Source
URL	457	Phrases occurring in the URL
OrigURL	495	Phrases occurring in the URL of the image
AncURL	472	Phrases occurring in the anchor text
Alt	111	Phrases occurring in the alternative text
Caption	19	Phrases occurring in the caption terms

Table 5.19 gives the classification results on the Advert data set obtained by SVM, MKL, and LMKL. We see that SVM (All) is significantly more accurate than the best SVM with single feature representation, namely SVM (AncURL), and uses significantly fewer support vectors. MKL has comparable classification accuracy to SVM (All) and the difference between the number of support vectors is not significant. LMKL with softmax/sigmoid gating has comparable accuracy to MKL and SVM (All). LMKL with sigmoid gating stores significantly fewer support vectors than SVM (All). The average kernel weights and the average number of active kernels on the Advert data set are given in Table 5.20. Different from the MultiFeat-SL data set, LMKL uses approximately the same number of or more kernels compared to MKL

on this data set (On one of the ten folds, MKL chooses five and on the remaining nine folds, it chooses four kernels, leading to an average of 4.1).

Table 5.19. Classification results on the ADVERT data set.

Method	Test Accuracy	Support Vector
SVM (URL)	94.67±0.24	83.32± 1.89
SVM (ORIGURL)	92.04±0.26	96.16± 0.51
SVM (ANCURL)	95.45±0.31	64.90± 5.41
SVM (ALT)	89.64±0.38	87.73± 1.17
SVM (CAPTION)	86.60±0.09	96.65± 0.42
SVM (ALL)	96.43±0.24	41.99± 1.76
MKL	96.32±0.50	35.82± 4.35
LMKL (softmax)	95.78±0.46	41.72±11.59
LMKL (sigmoid)	96.72±0.46	34.40± 1.51
LMKL (5 ANCURL and sigmoid)	95.66±0.29	10.87± 1.07

Table 5.20. Average kernel weights and number of active kernels on the ADVERT data set.

						Number of
Method	URL	ORIGURL	ANCURL	ALT	CAPTION	Active Kernels
MKL	0.3073	0.1600	0.3497	0.1828	0.0003	4.10
LMKL (softmax)	0.3316	0.0160	0.6292	0.0172	0.0060	4.04
LMKL (sigmoid)	0.9918	0.9820	0.9900	0.9913	0.4027	4.96

When we combine multiple copies of linear kernels on the ANCURL representation with LMKL using the sigmoid gating model on the same representation (see Figure 5.9), we see that LMKL stores much fewer support vectors compared to the single-kernel SVM (ANCURL) without sacrificing from accuracy. But, as before on the MULTIFEAT-SL data set, we could not achieve the classification accuracy obtained by combining different representations with sigmoid gating. For example, LMKL with sigmoid gating and kernels over five different feature representations is significantly better than LMKL with sigmoid gating and five copies of the kernel over the ANCURL representation in terms of classification accuracy but the latter stores significantly fewer support vectors (see Table 5.19).

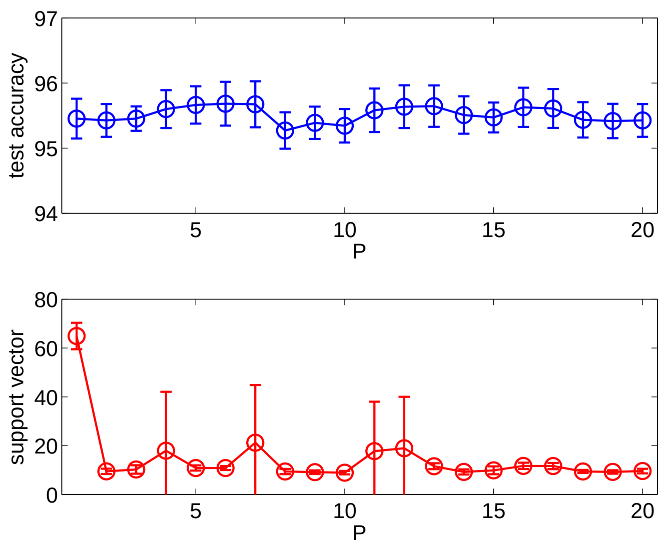


Figure 5.9. The average test accuracies and support vector percentages on the ADVERT data set obtained by LMKL with multiple copies of linear kernels and sigmoid gating on the ANCURL representation.

5.3.1.2. Combining Multiple Input Patches for Image Recognition Problems. For image recognition problems, only some parts of the images contain meaningful information and it is not necessary to examine the whole image in detail. Instead of defining kernels over the whole input image, we can divide the image into non-overlapping patches and use separate kernels in these patches. The kernels calculated on the parts with relevant information take nonzero weights and the kernels over the non-relevant patches are ignored. We use a low-resolution (simpler) version of the image as input to the gating model, which selects a subset of the high-resolution localized kernels. In such a case, it is not a good idea to use softmax gating in LMKL because softmax gating would choose one or very few patches and a patch by itself does not carry enough discriminative information.

We train SVMs with linear kernels calculated on the whole image in different resolutions. MKL and LMKL combine linear kernels calculated on each image patch. LMKL uses the whole image with different resolutions in the gating model (Gönen and Alpaydın, 2009a).

We perform experiments on the OLIVETTI data set, which consists of 10 different 64×64 grayscale images of 40 subjects. We construct a two-class data set by combining male subjects (36 subjects) into one class versus female subjects (four subjects) in another class. Our experimental methodology for this data set is slightly different: We select two images of each subject randomly and reserve these total 80 images as the test set. Then, we apply 8-fold cross-validation on the remaining 320 images by putting one image of each subject to the validation set at each fold. MKL and LMKL combine 16 linear kernels calculated on image patches of size 16×16.

Table 5.21 shows the results of MKL and LMKL combining kernels calculated over non-overlapping patches of face images. MKL achieves significantly higher classification accuracy than all single-kernel SVMs except in 32×32 resolution. LMKL with softmax gating has comparable classification accuracy to MKL and stores significantly fewer support vectors when 4×4 or 16×16 images are used in the gating model. This is mainly due to the normalization property of softmax gating that generally activates a single patch and ignores the others; this uses fewer support vectors but is not as accurate. LMKL with sigmoid gating significantly improves the classification accuracy over MKL by looking at the 8×8 images in the gating model and choosing a subset of the high-resolution patches.

Figure 5.10(b)-(c) show the combination weights found by MKL and sample face images stored as support vectors weighted with those. MKL uses the same weights over the whole input space and thereby the parts whose weights are nonzero are used in the decision process for all subjects. Figure 5.11 illustrates the example uses of LMKL with softmax and sigmoid gating. We see that the gating model activates important parts of each face image and these parts are used in the classifier with nonzero weights, whereas the parts whose gating model outputs are zero are not considered. That is, looking at the output of the gating model, we can skip processing the high-resolution versions of these parts. This can be considered similar to a selective attention mechanism whereby the gating model defines a saliency measure and drives a high-resolution "fovea"/"eye" to consider only regions of high saliency (Alpaydın, 1996). For example, if we use LMKL with softmax gating (see Figure 5.11(b)-(d)), the gating model gen-

Table 5.21. Classification results on the OLIVETTI data set.

Method	Test Accuracy	Support Vector
SVM ($\boldsymbol{x} = 4 \times 4$)	93.28±0.65	21.70±0.93
SVM ($\boldsymbol{x} = 8 \times 8$)	97.50±1.16	20.13±1.04
SVM ($\boldsymbol{x} = 16 \times 16$)	97.03±0.93	19.91±1.01
SVM ($\boldsymbol{x} = 32 \times 32$)	97.97±1.48	23.71±1.39
SVM ($\boldsymbol{x} = 64 \times 64$)	97.66±1.41	25.94±1.01
MKL	99.06±0.88	22.19±1.00
LMKL (softmax and $\boldsymbol{x}^{\mathcal{G}} = 4 \times 4$)	97.19±2.81	16.65±3.34
LMKL (softmax and $\boldsymbol{x}^{\mathcal{G}} = 8 \times 8$)	97.19±2.48	22.54±4.56
LMKL (softmax and $\boldsymbol{x}^{\mathcal{G}} = 16 \times 16$)	99.22±1.33	16.38±1.50
LMKL (sigmoid and $\boldsymbol{x}^{\mathcal{G}} = 4 \times 4$)	99.22±0.93	22.72±1.83
LMKL (sigmoid and $\boldsymbol{x}^{\mathcal{G}} = 8 \times 8$)	99.84±0.44	26.88±2.24
LMKL (sigmoid and $\boldsymbol{x}^{\mathcal{G}} = 16 \times 16$)	99.38±1.34	21.65±1.44

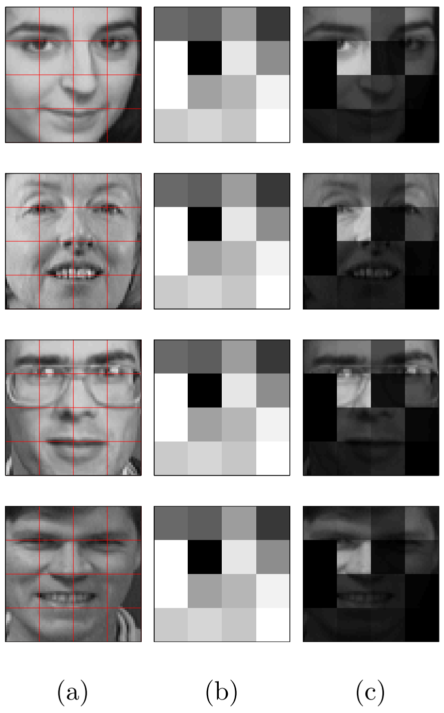


Figure 5.10. Example use of MKL on the OLIVETTI data set. (a) $\Phi_m(\boldsymbol{x}^m)$: features fed into kernels, (b) η_m: combination weights, and (c) $\eta_m\Phi_m(\boldsymbol{x}^m)$: features weighted with combination weights.

erally activates a single patch containing a part of eyes or eyebrows depending on the subject. This may not be enough for good discrimination and using sigmoid gating is more appropriate. When we use LMKL with sigmoid gating (see Figure 5.11(e)-(g)), multiple patches are given nonzero weights in a data-dependent way.

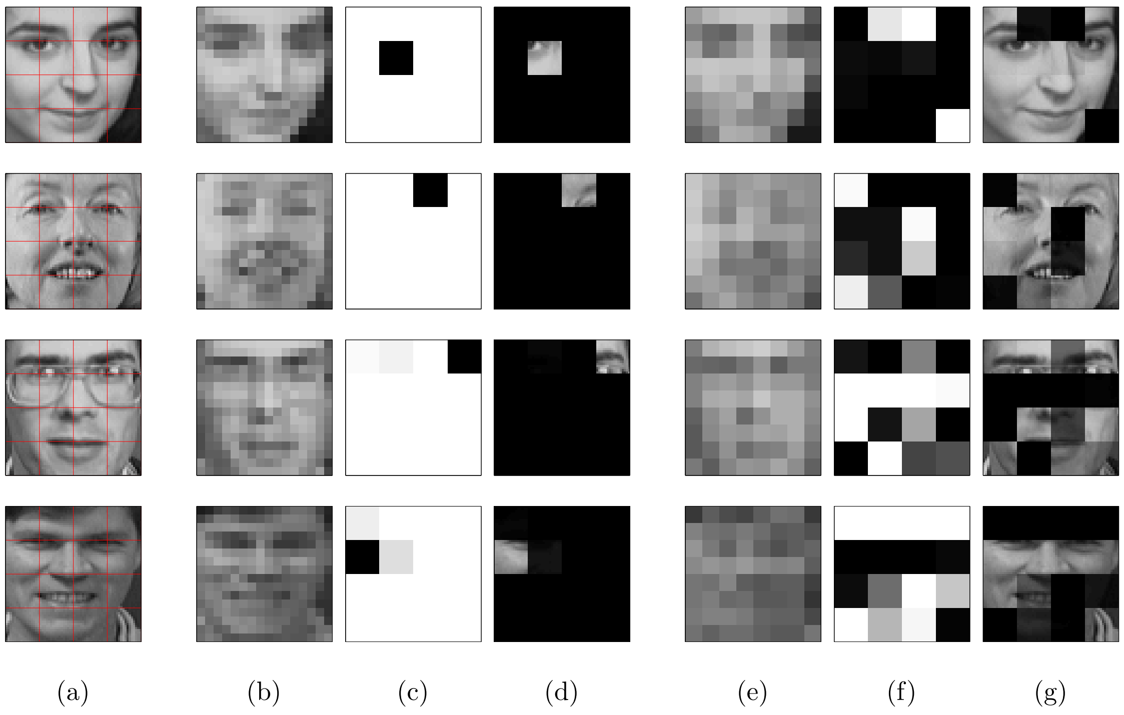


Figure 5.11. Example uses of LMKL on the OLIVETTI data set. (a) $\Phi_m(\boldsymbol{x}^m)$: features fed into kernels, (b) $\boldsymbol{x}^{\mathcal{G}}$: features fed into softmax gating, (c) $\eta_m(\boldsymbol{x}|\mathbf{V})$: softmax gating outputs, (d) $\eta_m(\boldsymbol{x}|\mathbf{V})\Phi_m(\boldsymbol{x}^m)$: features weighted with softmax gating outputs, (e) $\boldsymbol{x}^{\mathcal{G}}$: features fed into sigmoid gating, (f) $\eta_m(\boldsymbol{x}|\mathbf{V})$: sigmoid gating outputs, and (g) $\eta_m(\boldsymbol{x}|\mathbf{V})\Phi_m(\boldsymbol{x}^m)$: features weighted with sigmoid gating outputs.

Figure 5.12 gives the average kernel weights on the test set for MKL, LMKL with softmax gating, and LMKL with sigmoid gating. We see that MKL and LMKL with softmax gating use fewer high-resolution patches than LMKL with sigmoid gating.

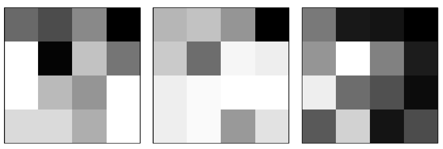


Figure 5.12. Average kernel weights on the OLIVETTI data set. (a) MKL, (b) LMKL with softmax gating on 16×16 resolution, and (c) LMKL with sigmoid gating on 8×8 resolution.

We can generalize this idea even further: Let us say that we have a number of information sources that are costly to extract or process, and a relatively simpler one. In such a case, we can feed the simple representation to the gating model and feed the costly representations to the actual kernels and train LMKL. The gating model then chooses a costly representation only when it is needed and chooses only a subset of the costly representations. Note that the representation used by the gating model does not need to be very precise, because it does not do the actual decision, but only chooses the representation(s) that do the actual decision.

5.3.2. Regression Experiments

We compare SVR and LMKL in terms of regression performance (i.e., mean square error) and model complexity (i.e., stored support vector percentage). We train SVRs with different kernels, namely the linear kernel and polynomial kernels up to fifth degree. LMKL combines these five kernels with both softmax and sigmoid gating.

We perform experiments on the Concrete Compressive Strength (CONCRETE) data set and the Wine Quality (WHITEWINE) data set from the UCI Machine Learning Repository. ϵ is selected from {1, 2, 4, 8, 16} for the CONCRETE data set and {0.08, 0.16, 0.32, 0.64, 1.28} for the WHITEWINE data set.

Table 5.22 lists the regression results on the CONCRETE data set obtained by SVR and LMKL. We see that both LMKL with softmax gating and LMKL with sigmoid gating are significantly more accurate than all of the single-kernel SVRs. LMKL with softmax gating uses k_L, k_P ($q = 4$), and k_P ($q = 5$) with relatively higher weights but LMKL with sigmoid gating uses all of the kernels with significant weights (see Table 5.23). When we combine multiple copies of the linear kernel using the softmax gating model, (shown in Figure 5.13), we see that LMKL does not overfit and we get significantly lower error than the best single-kernel SVR (k_P and $q = 3$). For example, LMKL with five copies of k_L and softmax gating gets significantly lower error than SVR (k_P and $q = 3$) and stores significantly fewer support vectors.

Table 5.22. Regression results on the CONCRETE data set.

Method	MSE	Support Vector
SVR (k_L)	120.61±2.15	44.31±3.46
SVR (k_P and $q = 2$)	92.57±4.19	36.22±1.24
SVR (k_P and $q = 3$)	58.32±3.66	73.16±1.21
SVR (k_P and $q = 4$)	63.83±9.58	52.52±2.40
SVR (k_P and $q = 5$)	61.26±5.31	52.17±2.23
LMKL (softmax)	44.80±6.33	64.28±4.02
LMKL (sigmoid)	48.18±5.22	49.20±2.05
LMKL (5 k_L and softmax)	53.14±6.42	34.93±8.45

Table 5.23. Average kernel weights and number of active kernels on the CONCRETE data set.

	k_L	k_P	k_P	k_P	k_P	Number of
Method		$q = 2$	$q = 3$	$q = 4$	$q = 5$	Active Kernels
LMKL (softmax)	0.1495	0.0091	0.0117	0.0951	0.7346	4.52
LMKL (sigmoid)	0.6675	0.8176	0.9962	0.9721	0.9989	4.68

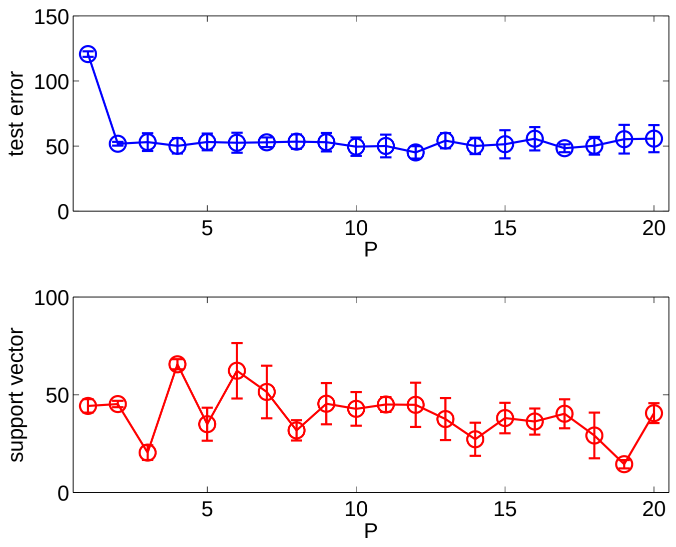


Figure 5.13. The average test mean square errors and support vector percentages on the CONCRETE data set obtained by LMKL with multiple copies of linear kernels and softmax gating.

Table 5.24 lists the regression results on the WhiteWine data set obtained by SVR and LMKL. We see that both LMKL with softmax gating and LMKL with sigmoid gating obtain significantly less error than SVR (k_L), SVR (k_P and $q = 2$), and SVR (k_P and $q = 3$), and have comparable error to SVR (k_P and $q = 4$) and SVR (k_P and $q = 5$) but store significantly fewer support vectors than all single-kernel SVRs. Even if we do not decrease the error, we learn computationally simpler models by storing much fewer support vectors. We see from Table 5.25 that LMKL with softmax gating assigns relatively higher weights to k_L, k_P ($q = 3$), and k_P ($q = 5$), whereas LMKL with sigmoid gating uses the polynomial kernels nearly everywhere in the input space and the linear kernel for some of the test instances.

Table 5.24. Regression results on the WhiteWine data set.

Method	MSE	Support Vector
SVR (k_L)	0.59±0.00	66.83± 0.57
SVR (k_P and $q = 2$)	0.54±0.01	66.22± 0.67
SVR (k_P and $q = 3$)	0.54±0.00	66.14± 1.13
SVR (k_P and $q = 4$)	0.52±0.01	66.55± 1.03
SVR (k_P and $q = 5$)	0.52±0.01	66.27± 1.24
LMKL (softmax)	0.52±0.01	18.66±13.41
LMKL (sigmoid)	0.51±0.00	38.29± 2.34

Table 5.25. Average kernel weights and number of active kernels on the WhiteWine data set.

	k_L	k_P	k_P	k_P	k_P	Number of
Method		$q = 2$	$q = 3$	$q = 4$	$q = 5$	Active Kernels
LMKL (softmax)	0.2238	0.0302	0.1430	0.0296	0.5733	1.05
LMKL (sigmoid)	0.5956	0.9698	0.9978	0.9849	0.9929	4.58

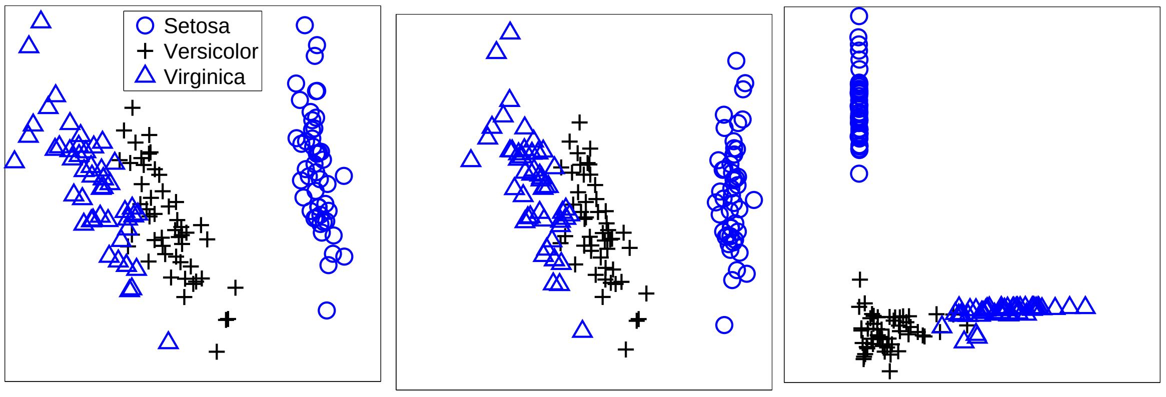
5.4. Local Projection Kernels Experiments

In this section, we evaluate the performance of our proposed method LPK of Chapter 4 on visualization and classification tasks using benchmark data sets. To compare, we use the MATLAB implementation of LFDA (Sugiyama, 2007) with default parameters.

5.4.1. Data Visualization

We compare PCA, LFDA, and our proposed LPK for data visualization on small benchmark data sets, namely Iris, Thyroid Disease, Letter Recognition, and Image Segmentation from the UCI Machine Learning Repository. On these multiclass data sets, we merge certain classes, as done by Sugiyama (2007), to obtain multimodal two-class problems. In PCA and LFDA methods, we extract two dimensions by using the first two principal directions. In LPK method using SVM with the linear kernel, we use two regions ($P = 2$) with the softmax gating model and project data points to one dimension ($R_m = 1$) in each region.

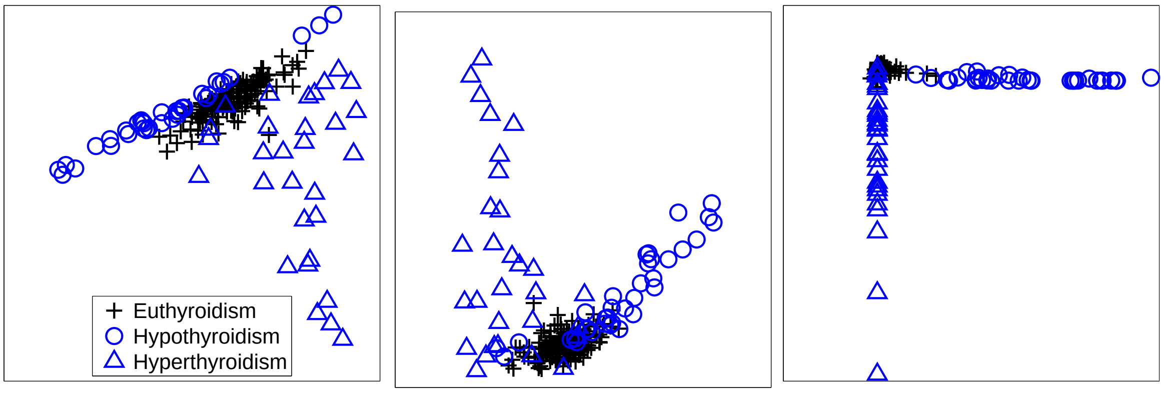
On the Iris data set, we combine Setosa and Virginica into a single class to obtain multimodality. Figure 5.14 shows the two-dimensional projected feature spaces found by each method. Both PCA and LFDA preserve within-class modality but could not achieve a clear between-class separation. However, our proposed LPK achieves a clear between-class separation while preserving within-class modality.



(a) PCA (b) LFDA (c) LPK

Figure 5.14. Data visualization on the Iris data set.

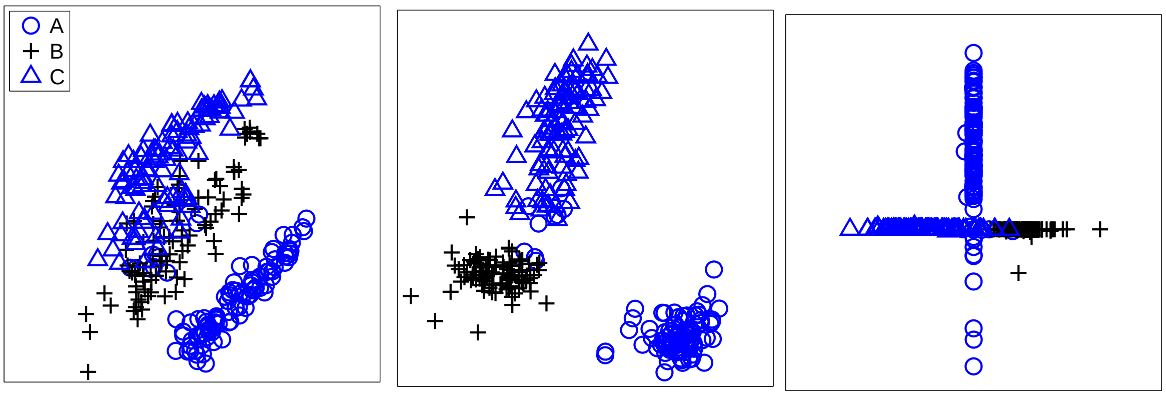
On the Thyroid Disease data set, we merge Hypothyroidism and Hyperthyroidism classes into one class. As we can see from Figure 5.15, all three methods obtain similar results but LPK has better separation between within-class modalities.



(a) PCA (b) LFDA (c) LPK

Figure 5.15. Data visualization on the Thyroid Disease data set.

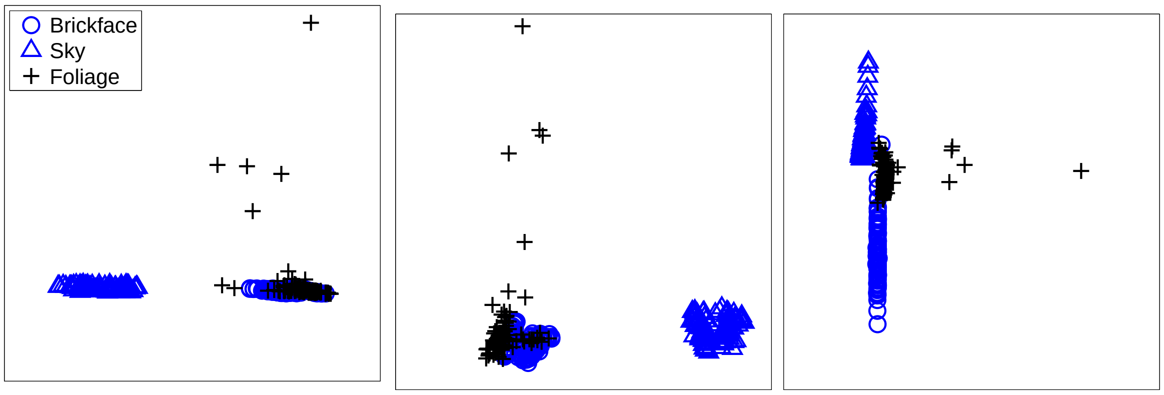
On the Letter Recognition data set, we construct a two-class data set by combining ‘A’ and ‘C’ letters into one class versus ‘B’ letter in another class. LFDA achieves a good separation between clusters, whereas PCA is not able to separate the samples from ‘B’ and ‘C’ letters (see Figure 5.16). LPK also achieves good separation between different classes but it could not separate letters ‘A’ and ‘C’ as well as LFDA. This is mainly because of the discriminative nature of LPK, the main goal is to separate different classes rather than preserving multimodality in one class.



(a) PCA (b) LFDA (c) LPK

Figure 5.16. Data visualization on the Letter Recognition data set.

On the Image Segmentation data set, we combine Brickface and Sky classes into one class and treat Foliage as another class. Figure 5.17 shows that PCA and LFDA are not able to separate Brickface and Foliage classes, whereas LPK obtains three different clusters for each class while maintaining a good between-class separation.



(a) PCA (b) LFDA (c) LPK

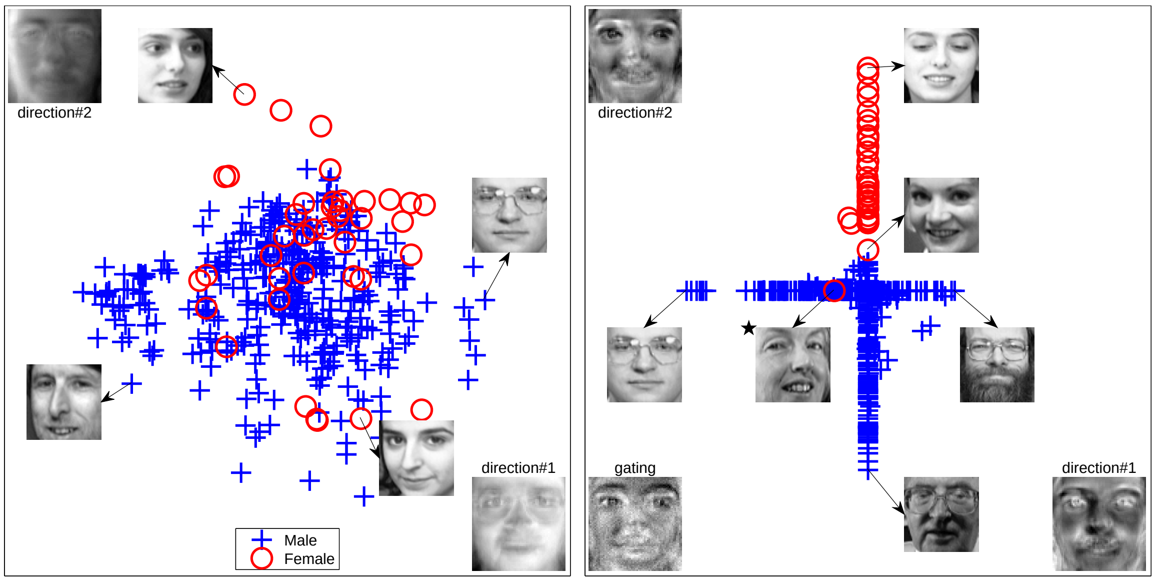
Figure 5.17. Data visualization on the IMAGE SEGMENTATION data set.

5.4.2. Face Recognition

We also compare PCA and LPK on the OLIVETTI face recognition data set in order to see the performance of LPK in a real-life scenario with a very high dimensional feature space. The OLIVETTI data set consists of 10 different 64×64 grayscale images of 40 subjects. We construct a two-class data set by combining male subjects (36 subjects) into one class versus female subjects (four subjects) in another class. In both methods, we project data points to a two-dimensional space. PCA extracts these two dimensions by using the first two principal directions, and LPK using SVM with the linear kernel divides the original feature space into two regions ($P = 2$) with the softmax gating model and projects data points to one dimension ($R_m = 1$) in each region.

Figure 5.18(a) illustrates the projection obtained by PCA. We can see that PCA is not able to separate classes due to its unsupervised nature. Eigenfaces obtained from the first two principal eigenvectors are also shown on the two corners and they look like two male subjects.

LPK finds a better two-dimensional projected space as shown in Figure 5.18(b). LPK is able to achieve a nearly perfect separation between classes except a single image. We also produce a face image from the gating model parameters, $\{\boldsymbol{v}_1, \boldsymbol{v}_2\} \in$



(a) PCA (b) LPK

Figure 5.18. Data visualization on the OLIVETTI data set. (a) PCA: Two eigenface images obtained from the first two principal eigenvectors are shown in the corners. (b) LPK ($P = 2$ and $R_m = 1$): The face images obtained from the local projection matrices, $\mathbf{W}_1$ and $\mathbf{W}_2$, are shown in the corners. Only the image marked $\star$ is misclassified.

$\mathbb{R}^{4096\times 1}$, in order to see which features are important when dividing the input space into local regions. If we look at the face image produced, we can see that the gating model puts more emphasis on eyes, eyebrows, nose, and mouth to assign the weights to the local projection spaces for a given data instance. The face image obtained from the first local projection matrix, $\mathbf{W}_1 \in \mathbb{R}^{4096\times 1}$, is very much like a male subject with relatively higher weights on eyebrows and nose. The face image of the other local projection matrix, $\mathbf{W}_2 \in \mathbb{R}^{4096\times 1}$, looks like a female subject with relatively higher weights on eyes and mouth. LPK identifies the important parts of the face images without using any prior information while trying to optimize the separation between classes in a supervised manner.

5.4.3. Classification Experiments

We evaluate the performance of PCA, LFDA, and LPK on classification tasks using large benchmark data sets. Table 5.26 lists the properties of the data sets. WAVEFORM from the UCI repository is selected due to its multimodal structure and the first two classes are combined into a single one. USPS-EO (USPS-SL) are generated from USPS data set (16×16 grayscale digit images) by combining even (small: '0' - '4') numbers and odd (large: '5' - '9') numbers into different classes.

Table 5.26. Classification data sets used in the experiments.

Data Set	Dimensionality	Number of Instances
WAVEFORM	21	1500
USPS-EO	256	1500
USPS-SL	256	1500

We also train SVMs after reducing dimensionality with PCA or LFDA using the same kernel in GPK and LPK (the linear kernel in our experiments). Note that GPK algorithm discussed in Section 4.1 is equivalent to LPK with $P = 1$.

On the WAVEFORM data set (see Figure 5.19), SVM trained after PCA and LFDA obtains nearly the same average accuracy results (around 89 per cent) after two dimensions. GPK achieves similar accuracy results with only one dimension. If we use local projection matrices with LPK ($P = 2$ or $P = 3$), the average classification accuracy increases to 92 per cent using few dimensions. Because dimensionality reduction is done separately in different regions, we can work with much fewer dimensions attaining significantly higher accuracy. For example, when $R_m = 2$ or $R_m = 3$, LPK ($P = 2$ or $P = 3$) stores significantly fewer support vectors than SVM trained after PCA and LFDA while achieving significantly higher accuracy. Fitting a simpler model while attaining higher test accuracy is a clear indication of better generalization. SVM without any dimensionality reduction obtains 88.34 per cent average accuracy.

On the USPS-EO data set (see Figure 5.20), SVM trained after LFDA obtains an average accuracy around 79 per cent for all dimension values tried. However, SVM

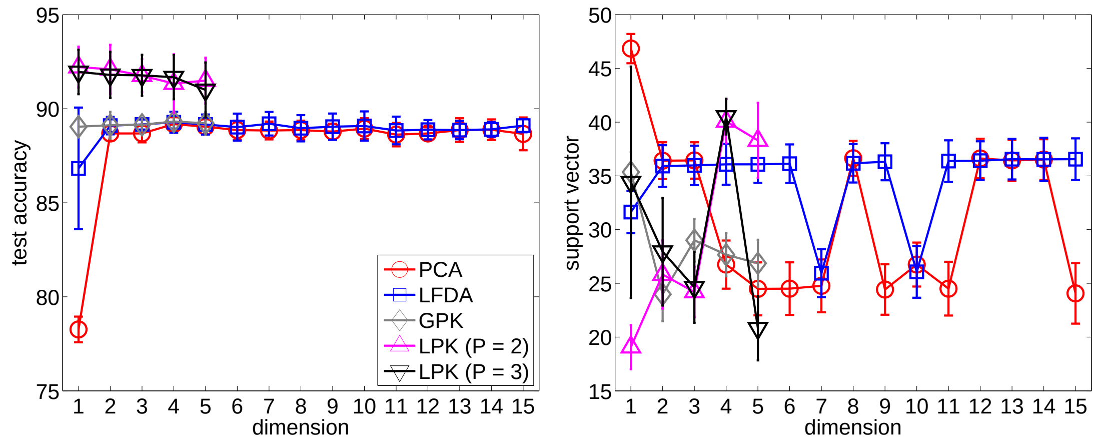


Figure 5.19. Classification results on the WAVEFORM data set.

trained after PCA obtains better average accuracies after five dimensions and 86.10 per cent average accuracy with 15 dimensions. GPK and LPK ($P = 3$) obtains more than 87 and 90 per cent average accuracy, respectively, for all dimension values tried. GPK and LPK achieve significantly higher accuracies and store significantly fewer support vectors than SVM trained after PCA for all configurations. SVM without any dimensionality reduction obtains 87.58 per cent average accuracy.

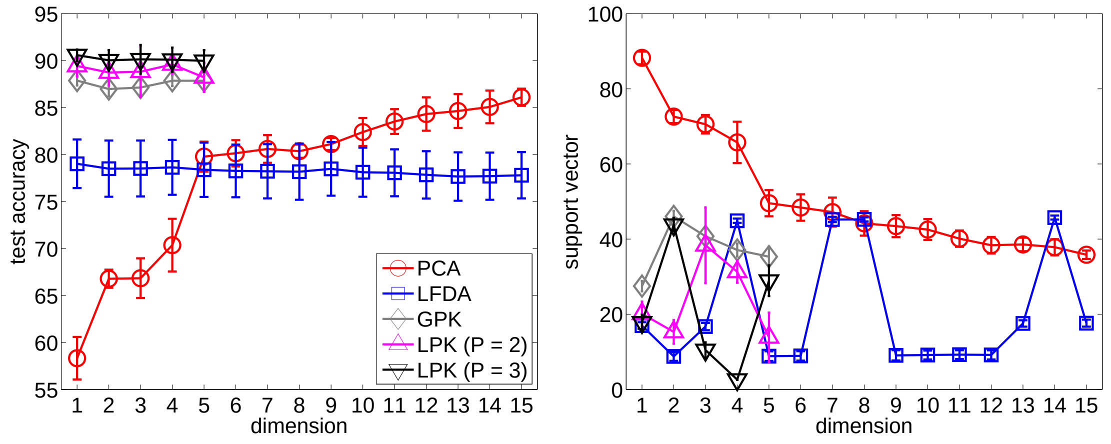


Figure 5.20. Classification results on the USPS-EO data set.

On USPS-SL (see Figure 5.21), SVM trained after PCA has more than 70 per cent average accuracy after 14 dimensions, whereas SVM trained after LFDA gets around 68 per cent average accuracy. GPK achieves average accuracy more than 75 per cent. When we use local projection matrices ($P = 2$ or $P = 3$), the average accuracy increases to more than 86 per cent. LPK achieves 11 per cent higher accuracy than GPK and stores only 5 per cent of training instances as support vectors. SVM without any dimensionality reduction obtains 76.36 per cent average accuracy.

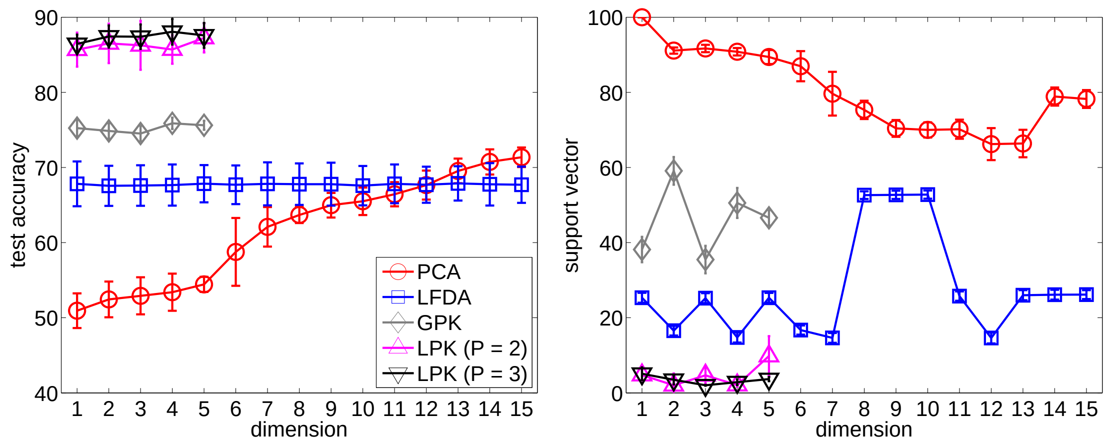


Figure 5.21. Classification results on the USPS-SL data set.

We also compare the classification performances of these methods on Olivetti (see Figure 5.22). We use a different methodology for this data set. We select two images of each subject randomly and reserve these total 80 images as the test set. Then, we apply 8-fold cross-validation on the remaining 320 images by putting one image of each subject to the validation set at each fold. In order to get rid of singularity problems in LFDA method, we project data instances into a 100-dimensional space with PCA before applying LFDA. SVM trained after PCA could not achieve more than 96 per cent average accuracy, whereas SVM trained after LFDA gets around 98 per cent average accuracy. GPK achieves average accuracy more than 98 per cent with four and five dimensions. LPK ($P = 2$ or $P = 3$) achieves more than 98 per cent average accuracy after two dimensions ($R_m \geq 2$). For example, LPK ($P = 2$ and $R_m = 4$) obtains 99.69 per cent average accuracy. LPK stores nearly the same amount of support vectors as SVM trained after LFDA but achieves higher average accuracy. SVM without any dimensionality reduction obtains 99.06 per cent average accuracy.

5.4.4. Convergence Analysis

We perform convergence analysis of LPK on Waveform and Olivetti data sets. We train LPK ($P = 2$, $R_m = 1$, and $C = 100$) with the linear kernel for 25

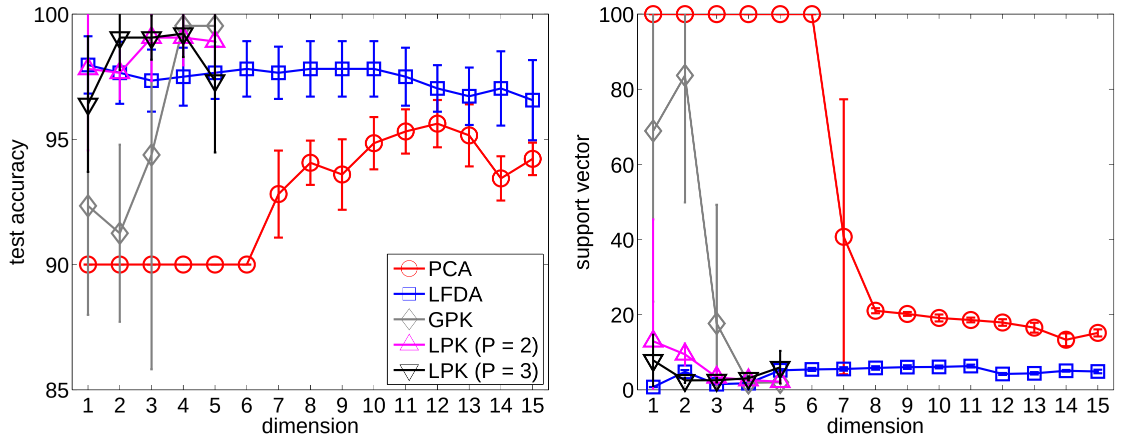


Figure 5.22. Classification results on the OLIVETTI data set.

iterations and record the objective function value, training and test set accuracies, and the percentage of support vectors at each iteration.

Figure 5.23 shows that LPK converges on WAVEFORM data set after five iterations. If we use the stopping condition based on the objective function value with $\tau = 0.01$ or 0.001, LPK stops respectively after 9 and 13 iterations (shown with a square and a circle).

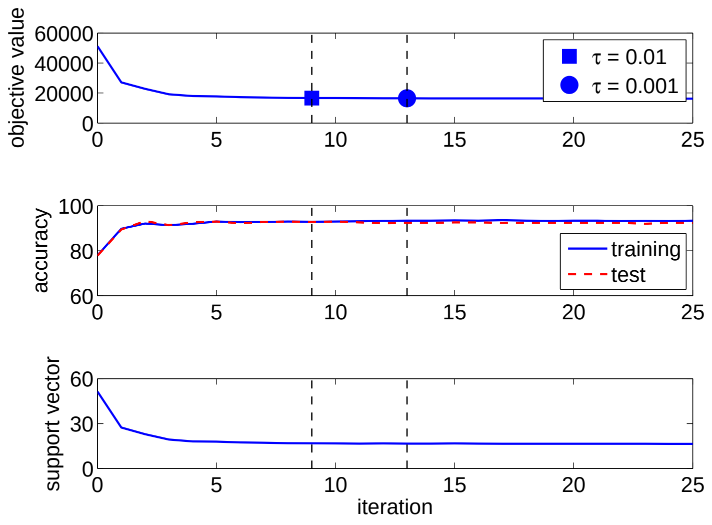


Figure 5.23. Convergence analysis of LPK on the WAVEFORM data set.

A similar behavior is also seen on the OLIVETTI data set (see Figure 5.24). We see that even $\tau = 0.01$ is too conservative. Note that on both data sets, LPK does not overfit even if we allow all 25 iterations.

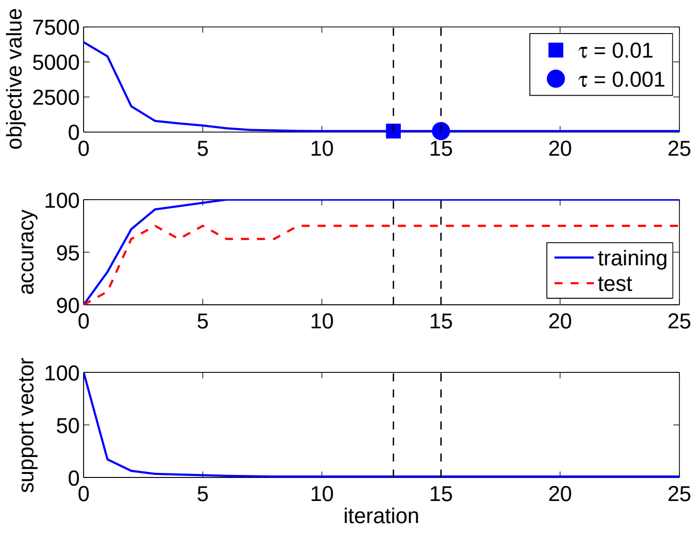


Figure 5.24. Convergence analysis of LPK on the OLIVETTI data set.

6. CONCLUSIONS AND FUTURE WORK

In recent years, several MKL methods have been proposed in the machine learning literature. Different kernels correspond to different notions of similarity and MKL can be used to combine them. It can also be used to integrate different inputs coming from different representations, possibly from different sources or modalities, by combining kernels calculated on these representations.

This thesis contains a number of extensions to the original MKL framework, together with experimental results that support their utility on benchmark data sets from the UCI Machine Learning Repository as well as several image recognition and bioinformatics data sets.

6.1. Contributions of the Thesis

This thesis introduces a *regularized multiple kernel learning* framework extending the MKL formulation of Bach *et al.* (2004). The newly introduced regularization parameters should not be set equal; our experimental results show that they have an effect on accuracy and this thesis proposes to use RSM to have a regularized variant of MKL to search for the best parameter set using validation data. Our proposed method, RMKL, is tested on several bioinformatics and digit recognition data sets and we see that it eliminates some of the kernel functions or decreases the support vector count, sometimes also improves accuracy, but never sacrifices from accuracy.

Optimizing the regularization parameter of each kernel allows us to obtain more robust decision functions for the classification task at hand. Kernels that do not help increase the classification accuracy are pruned by selecting their regularization parameters accordingly, obtaining smoother discriminants. Eliminating some of the kernels directly or decreasing the number of stored support vectors reduces the testing time for new instances. In an application where there is a single source and multiple kernels, determining which kernels are favored and which are not needed gives us information, indicating which notions of similarity are valid. When different kernels use information from different sources, pruning kernels corresponds to pruning redundant information sources; after all, not all sources may be necessary.

This thesis also proposes a *cost-conscious multiple kernel learning* framework to include the cost of kernel computations and data acquisition/generation into the MKL mathematical model of Bach *et al.* (2004). We present results for two sets of experiments on benchmark data sets to combine different kernels on the same data representation and to combine different data representations with the same kernel. The results show that incorporating a cost factor into the model enables us to use only the necessary kernels, avoiding costly kernel computations and input generation for some data representations in the testing phase, when possible. The cost of a kernel depends on the time/space complexity of the kernel implementation (in software or hardware) and the cost of sensing the input representation and manipulating it.

The cost-conscious MKL variant is also tested on two bioinformatics applications described from CYGD. Similar to the results obtained on the UCI benchmark data sets, the cost-conscious variant of MKL helps us trade off the contribution of a kernel to accuracy with its complexity and can eliminate expensive data representations/kernels when possible. By using cost-conscious MKL in bioinformatics applications, we show that one can select a kernel combination that enables us to get rid of the costlier data representations (generally obtained through additional experimental processes) and evaluating the expensive kernel functions.

The main contribution of this thesis is the formulation of a *localized multiple kernel learning* framework for kernel-based algorithms. Our proposed algorithm, LMKL, has two main ingredients: the gating model that assigns weights to kernels for a data instance, and, the kernel-based learning algorithm that uses the locally combined kernel. The training of these two components are coupled and the parameters of both components are optimized together by using a two-step alternating optimization procedure. We derive the learning algorithm for three different gating models (softmax, sigmoid, and Gaussian) and apply LMKL to four different machine learning problems (binary classification, regression, multiclass classification, and one-class classification).

We perform experiments on several binary classification and regression problems. We compare the empirical performance of LMKL with single-kernel SVM and SVR as well as MKL. For classification problems that use different feature representations, LMKL is able to construct better classifiers than MKL by combining the kernels on these representations locally. In our experiments, LMKL achieves higher average test accuracies and stores fewer support vectors compared to MKL. If the combined feature representations are complementary and do not contain redundant information, the sigmoid gating model should be selected instead of softmax gating, in order to have the possibility of using more than one representation. We also see that, as expected, combining heterogeneous feature representations is more advantageous than combining multiple copies of the same representation. For image recognition problems, LMKL identifies the relevant parts of each input image separately by using the gating model as a saliency detector on the kernels calculated on the image patches, and we see that LMKL obtains better classification results than MKL on a gender recognition task using face images. For regression problems, LMKL improves the performance by reducing the mean square error significantly or storing significantly fewer support vectors. Different from MKL methods that use global kernel weights, LMKL can combine multiple copies of the same kernel. We show that even if we provide more kernels than needed, LMKL uses only as many support vectors as required and does not overfit.

The computational complexity of LMKL depends on the complexity of the canonical kernel machine solver used in the main loop and the number of iterations before convergence. For example, solving a canonical SVM optimization problem has $\mathcal{O}(N^2)$ space and $\mathcal{O}(N^3)$ time complexity, whereas gradient calculations for the gating model parameters have a smaller time complexity. In practice, we see convergence in five to 20 iterations using gradient-descent with Armijo's rule as the line search method. In two-step alternating optimization algorithm of LMKL, instead of using gradient-descent to update the gating model parameters, one can use a more complex procedure such as simulated annealing. This would increase time complexity but may help us find better solutions.

We also introduce a supervised and localized dimensionality reduction method that trains *local projection kernels* coupled with a kernel-based learning algorithm. Our proposed method, LPK, has three main ingredients: the gating model that assigns weights to projection matrices for a data instance, the local projection matrices that perform dimensionality reduction separately in each region constructed by the gating model, and, the kernel-based learning algorithm that combines these locally constructed features. The training of these three components are coupled, are all supervised, and the parameters of components are optimized together by using an alternating optimization scheme. The result of combining these three components is a local projection kernel that performs locality preserving projection while considering the accuracy of the discriminant formed using such kernels. For binary classification tasks, the mathematical details of our proposed framework with the softmax gating model are given. We discuss how the same derivation can be extended to regression estimation and novelty detection problems.

Our proposed method, LPK, is tested and compared with two other algorithms, PCA and LFDA, for data visualization and classification tasks on benchmark data sets. On visualization tasks, LPK is able to maintain the multimodality of a class by placing clusters of the same class on the same side of the hyperplane while preserving a separation between them. This property is a direct result of using a gating model in LPK. On classification tasks, LPK achieves better results than PCA and LFDA by attaining both higher test accuracy and storing fewer support vectors, due to the coupled optimization of the discriminant and the local projection matrices used in dimensionality reduction.

6.2. Future Work

As an extension to the proposed regularization framework of RMKL, RSM on cross-validation performance can also be used in other model selection settings. For example, the regularization parameter of kernel machines and the spread parameter of the Gaussian kernel can be selected using RSM. By doing this, we can get rid of the costly grid search procedure and can select arbitrary parameter values (grid search can only select from a list with predefined precision).

In the LMKL framework, the most time-consuming step is solving the canonical kernel machine with the locally combined kernel matrix. In order to reduce the overall time complexity, we can use the least squares formulations of canonical kernel machines. Such a procedure has an analytical solution that requires inverting the locally combined kernel matrix. Because the kernel matrix changes gradually when the gating model parameters change, the inverse calculated in the previous iteration can be used to easily calculate the inverse of the current kernel matrix.

The LMKL framework uses a parametric gating model that restricts the method to use a specified number of kernel functions. Both the gating model and the inner kernel machines are discriminative models. The number of kernels to be combined should be determined before training and this is the main restriction of the LMKL method. A Bayesian reformulation of the gating model in LMKL with infinite Dirichlet mixtures can both select the number of kernels to be combined and learn to combine these kernels. This would be a hybrid formulation of discriminative and generative models applied to kernel combination. Instead of regularizing LMKL using a statistical cross-validation procedure, infinite Dirichlet mixtures could select the model complexity while training. The classifier in the inner loop of LMKL could also be a generative model like Gaussian processes and we can obtain a fully generative formulation of LMKL.

If the dimensionality of the feature representation used in the gating models of LMKL and LPK is very high, we may need to optimize a very large number of parameters. Instead, we can also reduce the dimensionality of the gating representation and learn the gating model parameters in this new projected space.

APPENDIX A: STATISTICAL TESTS USED

In this chapter, we review the statistical tests used for comparing the algorithms given the significance level, α.

A.1. 5×2 cv Paired F Test

. In 5×2 cross-validation, we perform five replications of twofold cross-validation. In each replication, the training set is divided into two equal-sized sets. Let $p_i^{(j)}$ be the difference between the performance values of the two learners on fold $j = 1, 2$ of replication $i = 1, 2, \ldots, 5$. The average on replication i is $\bar{p}_i = (p_i^{(1)} + p_i^{(2)})/2$, and the estimated variance $s_i^2 = (p_i^{(1)} - \bar{p}_i)^2 + (p_i^{(2)} - \bar{p}_i)^2$. If $p_i^{(1)}/\sigma \sim \mathcal{Z}$, then $(p_i^{(j)})^2/\sigma^2 \sim \chi_1^2$ and their sum is chi-square with ten degrees of freedom:

$$N = \frac{\sum\limits_{i=1}^{5}\sum\limits_{j=1}^{2}(p_i^{(j)})^2}{\sigma^2} \sim \chi_{10}^2$$

and assuming each of s_i^2 to be independent, their sum is chi-square with five degrees of freedom:

$$M = \frac{\sum\limits_{i=1}^{5} s_i^2}{\sigma^2} \sim \chi_5^2.$$

M and N divided by their respective degrees of freedom is F distributed with ten and five degrees of freedom (Alpaydın, 1999):

$$f = \frac{N/10}{M/5} = \frac{\sum\limits_{i=1}^{5}\sum\limits_{j=1}^{2}(p_i^{(j)})^2}{\sigma^2 \sum\limits_{i=1}^{5} s_i^2} \sim F_{10,5}.$$

The 5×2 cv paired F test accepts the hypothesis that two learners have the same performance at significance level, α, if this value is less than $F_{\alpha,10,5}$.

A.2. Wilcoxon's Signed Rank Test

The Wilcoxon's signed-rank test is a non-parametric test for comparing the results of two learners on several data sets (Wilcoxon, 1945). It ranks the differences between the performance values of two learners for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences.

Let p_i be the difference between the performance values of the two learners on data set $i = 1, 2, \ldots, M$. The differences are ranked according their absolute values. Let R^+ be the sum of ranks for the data sets on which the second learner outperformed the first. Let R^- be the sum of ranks for the data sets on which the first learner outperformed the second. Ranks of $p_i = 0$ are split evenly between R^+ and R^-; if there is an odd number of ties, one is ignored:

$$R^+ = \sum_{p_i>0} \text{rank}(p_i) + \frac{1}{2}\sum_{p_i=0} \text{rank}(p_i)$$
$$R^- = \sum_{p_i<0} \text{rank}(p_i) + \frac{1}{2}\sum_{p_i=0} \text{rank}(p_i).$$

Let T be the minimum of the sums, $T = \min(R^+, R^-)$. We can look up the exact critical value of T for small M values. Or, we can use the following statistics:

$$z = \frac{T - M(M+1)/4}{\sqrt{M(M+1)(2M+1)/24}} \sim Z.$$

where z is distributed approximately normally. The Wilcoxon's signed-rank test rejects the hypothesis that two learners have the same performance at significance level, α, if this value is less than $Z_{\alpha/2}$.

APPENDIX B: DATA SETS USED

In this chapter, we list the data sets used in the experiments for comparing algorithms.

B.1. Benchmark Data Sets

Tables B.1 and B.2 summarize the benchmark classification and regression data sets used in the experiments.

Table B.1. Benchmark classification data sets used in the experiments.

Name	N	D	K
Advert	3279	1558	2
Banana	5300	2	2
Heart	270	13	2
Image Segmentation	2310	19	7
Ionosphere	351	34	2
Iris	150	4	3
Liverdisorder	345	6	2
Letter Recognition	20000	16	26
Pima	768	8	2
Ringnorm	7400	20	2
Sonar	208	60	2
Spambase	4601	57	2
Thyroid Disease	215	5	3
Twonorm	7400	20	2
Waveform	5000	21	3
Wdbc	569	30	2

B.2. Image Recognition Data Sets

MultiFeat is a digit recognition data set composed of six different data representations for 2000 handwritten numerals (see Table 5.5).

Table B.2. Benchmark regression data sets used in the experiments.

Name	N	D
CONCRETE	1030	8
MOTORCYCLE	133	1
WHITEWINE	4898	11

OLIVETTI is a face recognition data set consisting of 10 different 64×64 grayscale images of 40 subjects.

USPS is a digit recognition data set composed of 16×16 grayscale images of 11000 digits.

OPTDIGITS is a optical digit recognition data set consisting of 8×8 on pixel counts calculated on 32×32 bitmaps of 5620 digits.

PENDIGITS is a pen-based digit recognition data set composed of eight successive pen points of 10992 digits on two-dimensional coordinate system.

B.3. Bioinformatics Data Sets

ACCEPTORS and DONORS are human splice site detection data sets consisting of 3889 and 6246 data instances, respectively (Kulp *et al.*, 1996).

ARABIDOPSIS and VERTEBRATES are translation initiation site detection data sets containing 2048 and 13454 instances, respectively (Pedersen and Nielsen, 1997).

POLYADENYLATION is a polyadenylation signal prediction data set containing 9255 instances for human DNA and mRNA sequences (Liu *et al.*, 2003).

MEMBRANE and RIBOSOMAL are protein subcellular location prediction data sets consisting of 2318 and 1150 data instances, respectively (Mewes *et al.*, 2000).

Y1, Y2, ..., Y13 are protein function prediction data sets constructed for 13 top-level categories of 3588 proteins in CYGD (Mewes *et al.*, 2000).

S2783 is a protein stability data set that contains 2471 single-site mutations of 68 proteins extracted from the ProTherm database (Özen *et al.*, 2009).

REFERENCES

Alimoğlu, F. and E. Alpaydın, 1997, "Combining Multiple Representations and Classifiers for Pen-based Handwritten Digit Recognition", *Proceedings of the 4th International Conference on Document Analysis and Recognition.*

Alpaydın, E., 1996, "Selective Attention for Handwritten Digit Recognition", *Advances in Neural Information Processing Systems 8.*

Alpaydın, E., 1999, "Combined 5×2 cv F test for Comparing Supervised Classification Learning Algorithms", *Neural Computation*, Vol. 11, No. 8, pp. 1885–1892.

Argyriou, A., R. Hauser, C. A. Micchelli, and M. Pontil, 2006, "A DC-Programming Algorithm for Kernel Selection", *Proceedings of the 23rd International Conference on Machine Learning.*

Argyriou, A., C. A. Micchelli, and M. Pontil, 2005, "Learning Convex Combinations of Continuously Parameterized Basic Kernels", *Proceeding of the 18th Conference on Learning Theory.*

Bach, F., 2008, "Consistency of the Group Lasso and Multiple Kernel Learning", *Journal of Machine Learning Research*, Vol. 9, pp. 1179–1225.

Bach, F. R., 2009, "Exploring Large Feature Spaces with Hierarchical Multiple Kernel Learning", *Advances in Neural Information Processing Systems 21.*

Bach, F. R., G. R. G. Lanckriet, and M. I. Jordan, 2004, "Multiple Kernel Learning, Conic Duality, and the SMO Algorithm", *Proceedings of the 21st International Conference on Machine Learning.*

Bach, F. R., R. Thibaux, and M. I. Jordan, 2005, "Computing Regularization Paths for Learning Multiple Kernels", *Advances in Neural Information Processing Systems*

17.

Belkin, M. and P. Niyogi, 2002, “Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering”, *Advances in Neural Information Processing Systems 14.*

Ben-Hur, A. and W. S. Noble, 2005, “Kernel Methods for Predicting Protein-Protein Interactions”, *Bioinformatics*, Vol. 21, No. Suppl 1, pp. i38–46.

Bennett, K. P., M. Momma, and M. J. Embrechts, 2002, “MARK: A Boosting Algorithm for Heterogeneous Kernel Models”, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Bi, J., T. Zhang, and K. P. Bennett, 2004, “Column-Generation Boosting Methods for Mixture of Kernels”, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Blum, B., M. I. Jordan, D. Kim, R. Das, P. Bradley, and D. Baker, 2008, “Feature Selection Methods for Improving Protein Structure Prediction with Rosetta”, *Advances in Neural Information Processing Systems 20.*

Bousquet, O. and D. J. L. Herrmann, 2003, “On the Complexity of Learning the Kernel Matrix”, *Advances in Neural Information Processing Systems 15.*

Bredensteiner, E. J. and K. P. Bennett, 1999, “Multicategory Classification by Support Vector Machines”, *Computational Optimization and Applications*, Vol. 12, No. 1-3, pp. 53–79.

Capriotti, E., P. Fariselli, and R. Casadio, 2004, “A Neural-Network-Based Method for Predicting Protein Stability Changes upon Single Point Mutations”, *Bioinformatics*, Vol. 20 (Supplement 1), pp. i63–i68.

Chapelle, O., V. Vapnik, O. Bousquet, and S. Mukherjee, 2002, “Choosing Multiple Parameters for Support Vector Machines”, *Machine Learning*, Vol. 46, No. 1–3, pp. 131–159.

Cheng, J., A. Randall, and P. Baldi, 2006, "Prediction of Protein Stability Changes for Single-Site Mutations Using Support Vector Machines", *Proteins: Structure, Function, and Bioinformatics*, Vol. 62, pp. 1125–1132.

Christoudias, C. M., R. Urtasun, and T. Darrell, 2009, "Bayesian Localized Multiple Kernel Learning", Tech. rep., Electrical Engineering and Computer Sciences, University of California at Berkeley.

Conforti, D. and R. Guido, 2010, "Kernel Based Support Vector Machine via Semidefinite Programming: Application to Medical Diagnosis", *Computers and Operations Research*, Vol. 37, No. 8, pp. 1389–1394.

Cortes, C., M. Mohri, and A. Rostamizadeh, 2009, "L_2 Regularization for Learning Kernels", *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*.

Cortes, C., M. Mohri, and A. Rostamizadeh, 2010, "Learning Non-Linear Combinations of Kernels", *Advances in Neural Information Processing Systems 22*.

Crammer, K., J. Keshet, and Y. Singer, 2003, "Kernel Design Using Boosting", *Advances in Neural Information Processing Systems 15*.

Crammer, K. and Y. Singer, 2001, "On the Algorithmic Implementation of Multiclass Kernel-Based Vector Machines", *Journal of Machine Learning Research*, Vol. 2, pp. 265–292.

Cristianini, N. and J. Shawe-Taylor, 2000, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press.

Cristianini, N., J. Shawe-Taylor, A. Elisseef, and J. Kandola, 2002, "On Kernel-Target Alignment", *Advances in Neural Information Processing Systems 14*.

Damoulas, T. and M. A. Girolami, 2008, “Probabilistic Multi-Class Multi-Kernel Learning: On Protein Fold Recognition and Remote Homology Detection”, *Bioinformatics*, Vol. 24, No. 10, pp. 1264–1270.

Damoulas, T. and M. A. Girolami, 2009a, “Combining Feature Spaces for Classification”, *Pattern Recognition*, Vol. 42, No. 11, pp. 2671–2683.

Damoulas, T. and M. A. Girolami, 2009b, “Pattern Recognition with A Bayesian Kernel Combination Machine”, *Pattern Recognition Letters*, Vol. 30, No. 1, pp. 46–54.

Dayhoff, M. O., R. M. Schwartz, and B. C. Orcutt, 1978, “A Model of Evolutionary Change in Proteins”, *Atlas of Protein Sequence and Structure*, Vol. 5 (Supplement 3), pp. 345–358.

De Bie, T., L.-C. Tranchevent, L. M. M. van Oeffelen, and Y. Moreau, 2007, “Kernel-Based Data Fusion for Gene Prioritization”, *Bioinformatics*, Vol. 23, No. 13, pp. i125–132.

de Diego, I. M., J. M. Moguerza, and A. Muñoz, 2004, “Combining Kernel Information for Support Vector Classification”, *Proceedings of the 4th International Workshop Multiple Classifier Systems*.

Dehak, R., N. Dehak, P. Kenny, and P. Dumouchel, 2008, “Kernel Combination for SVM Speaker Verification”, *Proceedings of the Speaker and Language Recognition Workshop*.

Fisher, R. A., 1936, “The Use of Multiple Measurements in Taxonomic Problems”, *Annals of Eugenics*, Vol. 7 Part II, pp. 179–188.

Fung, G., M. Dundar, J. Bi, and B. Rao, 2004, “A Fast Iterative Algorithm for Fisher Discriminant Using Heterogeneous Kernels”, *Proceedings of the 21st International Conference on Machine Learning*.

Gehler, P. V. and S. Nowozin, 2008, “Infinite Kernel Learning”, Tech. rep., Max Planck Institute for Biological Cybernetics.

Girolami, M. and S. Rogers, 2005, “Hierarchic Bayesian Models for Kernel Learning”, *Proceedings of the 22nd International Conference on Machine Learning.*

Girolami, M. and M. Zhong, 2007, “Data Integration for Classification Problems Employing Gaussian Process Priors”, *Advances in Neural Processing Systems 19.*

Globerson, A. and S. Roweis, 2006, “Metric Learning by Collapsing Classes”, *Advances in Neural Information Processing Systems 18.*

Gönen, M. and E. Alpaydın, 2008, “Localized Multiple Kernel Learning”, *Proceedings of the 25th International Conference on Machine Learning.*

Gönen, M. and E. Alpaydın, 2009a, “Localized Multiple Kernel Learning for Image Recognition”, *NIPS Workshop on Understanding Multiple Kernel Learning Methods.*

Gönen, M. and E. Alpaydın, 2009b, “Multiple Kernel Learning Algorithms”, Tech. Rep. FBE-CMPE-05/2009-02, Boğaziçi University.

Gönen, M. and E. Alpaydın, 2009c, “Multiple Kernel Machines Using Localized Kernels”, *Supplementary Proceedings of the 4th IAPR International Conference on Pattern Recognition in Bioinformatics.*

Gönen, M. and E. Alpaydın, 2010a, “Cost-Conscious Multiple Kernel Learning”, *Pattern Recognition Letters*, Vol. 31, No. 9, pp. 959–965.

Gönen, M. and E. Alpaydın, 2010b, “Localized Multiple Kernel Regression”, *Proceedings of the 20th International Conference on Pattern Recognition.*

Gönen, M. and E. Alpaydın, 2010c, “Regularizing Multiple Kernel Learning Using Response Surface Methodology”, *submitted.*

Gönen, M. and E. Alpaydın, 2010d, “Supervised Learning of Local Projection Kernels”, *Neurocomputing*, Vol. 73, No. 10–12, pp. 1694–1703.

Gönen, M., A. G. Tanuğur, and E. Alpaydın, 2008, “Multiclass Posterior Probability Support Vector Machines”, *IEEE Transactions on Neural Networks*, Vol. 19, No. 1, pp. 130–139.

Grandvalet, Y. and S. Canu, 2003, “Adaptive Scaling for Feature Selection in SVMs”, *Advances in Neural Information Processing Systems 15*.

Gromiha, M. M., J. An, H. Kono, M. Oobatake, H. Uedaira, P. Prabakaran, and A. Sarai, 2000, “ProTherm, Version 2.0: Thermodynamic Database for Proteins and Mutants”, *Nucleic Acids Research*, Vol. 28, pp. 283–285.

He, J., S.-F. Chang, and L. Xie, 2008, “Fast Kernel Learning for Spatial Pyramid Matching”, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

He, X. and P. Niyogi, 2004, “Locality Preserving Projections”, *Advances in Neural Information Processing Systems 16*.

Hou, C., J. Wang, Y. Wu, and D. Yi, 2009, “Local Linear Transformation Embedding”, *Neurocomputing*, Vol. 72, pp. 2368–2378.

Hsu, C.-W. and C.-J. Lin, 2002, “A Comparison of Methods for Multi-Class Support Vector Machines”, *IEEE Transactions on Neural Networks*, Vol. 13, No. 2, pp. 415–425.

Hu, M., Y. Chen, and J. T.-Y. Kwok, 2009, “Building Sparse Multiple-Kernel SVM Classifiers”, *IEEE Transactions on Neural Networks*, Vol. 20, No. 5, pp. 827–839.

Huang, L. T., M. M. Gromiha, S. F. Hwang, and S. Y. Ho, 2006, “Knowledge Acquisition and Development of Accurate Rules for Predicting Protein Stability Change”, *Computational Biology and Chemistry*, Vol. 30, pp. 408–415.

Igel, C., T. Glasmachers, B. Mersch, N. Pfeifer, and P. Meinicke, 2007, "Gradient-Based Optimization of Kernel-Target Alignment for Sequence Kernels Applied to Bacterial Gene Start Detection", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 4, No. 2, pp. 216–226.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton, 1991, "Adaptive Mixtures of Local Experts", *Neural Computation*, Vol. 3, pp. 79–87.

Joachims, T., N. Cristianini, and J. Shawe-Taylor, 2001, "Composite Kernels for Hypertext Categorisation", *Proceedings of the 18th International Conference on Machine Learning.*

Kandola, J., J. Shawe-Taylor, and N. Cristianini, 2002, "Optimizing Kernel Alignment over Combinations of Kernels", *Proceedings of the 19th International Conference on Machine Learning.*

Kim, S.-J., A. Magnani, and S. Boyd, 2006, "Optimal Kernel Selection in Kernel Fisher Discriminant Analysis", *Proceedings of the 23rd International Conference on Machine Learning.*

Kloft, M., U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien, 2010, "Efficient and Accurate l_p-Norm Multiple Kernel Learning", *Advances in Neural Information Processing Systems 22.*

Krefsel, U., 1998, "Pairwise Classification and Support Vector Machines", Schölkopf, B., C. Burges, and A. Smola (eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT Press.

Kulp, D., D. Haussler, M. G. Reese, and F. H. Eeckman, 1996, "A Generalized Hidden Markov Model for the Recognition of Human Genes in DNA", *Proceedings of 4th International Conference on Intelligent Systems for Molecular Biology.*

Kuncheva, L. I., 2004, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-

Interscience.

Lanckriet, G. R. G., N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, 2002, "Learning the Kernel Matrix with Semidefinite Programming", *Proceedings of the 19th International Conference on Machine Learning*.

Lanckriet, G. R. G., N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, 2004a, "Learning the Kernel Matrix with Semidefinite Programming", *Journal of Machine Learning Research*, Vol. 5, pp. 27–72.

Lanckriet, G. R. G., T. de Bie, N. Cristianini, M. I. Jordan, and W. S. Noble, 2004b, "A Statistical Framework for Genomic Data Fusion", *Bioinformatics*, Vol. 20, No. 16, pp. 2626–2635.

Lanckriet, G. R. G., M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble, 2004c, "Kernel-Based Data Fusion and Its Application to Protein Function Prediction in Yeast", *Proceedings of the Pacific Symposium on Biocomputing*.

Lee, C. and M. Levitt, 1991, "Accurate Prediction of the Stability and Activity Effects of Site-directed Mutagenesis on a Protein Core", *Nature*, Vol. 352, pp. 448–451.

Lee, W.-J., S. Verzakov, and R. P. W. Duin, 2007, "Kernel Combination versus Classifier Combination", *Proceedings of the 7th International Workshop Multiple Classifier Systems*.

Lee, Y., Y. Lin, and G. Wahba, 2001, "Multicategory Support Vector Machines", Tech. Rep. 1043, Department of Statistics, University of Wisconsin.

Lewis, D. P., T. Jebara, and W. S. Noble, 2006a, "Nonstationary Kernel Combination", *Proceedings of the 23rd International Conference on Machine Learning*.

Lewis, D. P., T. Jebara, and W. S. Noble, 2006b, "Support Vector Machine Learning from Heterogeneous Data: An Empirical Analysis Using Protein Sequence and Structure", *Bioinformatics*, Vol. 22, No. 22, pp. 2753–2760.

Li, X., S. Lin, S. Yan, and D. Xu, 2008, “Discriminant Locally Linear Embedding with High-Order Tensor Data”, *IEEE Transactions on Systems, Man, and Cybernetics, PartB*, Vol. 38, pp. 342–352.

Lin, Y.-Y., T.-L. Liu, and C.-S. Fuh, 2009, “Dimensionality Reduction for Data in Multiple Feature Representations”, *Advances in Neural Processing Systems 21.*

Liu, H., H. Han, J. Li, and L. Wong, 2003, “An In-silico Method for Prediction of Polyadenylation Signals in Human Sequences”, *Proceedings of the 14th International Conference on Genome Informatics.*

Lodhi, H., C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, 2002, “Text Classification Using String Kernels”, *Journal of Machine Learning Research*, Vol. 2, pp. 419–444.

Longworth, C. and M. J. F. Gales, 2008, “Multiple Kernel Learning for Speaker Verification”, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing.*

Longworth, C. and M. J. F. Gales, 2009, “Combining Derivative and Parametric Kernels for Speaker Verification”, *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 17, No. 4, pp. 748–757.

Mayoraz, E. and E. Alpaydın, 1999, “Support Vector Machines for Multi-Class Classification”, Mira, E. J. and J. V. S. Andres (eds.), *Engineering Applications of Bio-Inspired Artificial Neural Networks*, pp. 833–842, Springer.

McFee, B. and G. Lanckriet, 2009, “Partial Order Embedding with Multiple Kernels”, *Proceedings of the 26th International Conference on Machine Learning.*

Mewes, H. W., D. Frishman, C. Gruber, B. Geier, D. Haase, A. Kaps, K. Lemcke, G. Mannhaupt, F. Pfeiffer, C. Schüller, S. Stocker, and B. Weil, 2000, “MIPS: A Database for Genomes and Protein Sequences”, *Nucleic Acid Research*, Vol. 28, pp. 37–40.

Micchelli, C. A. and M. Pontil, 2005, "Learning the Kernel Function via Regularization", *Journal of Machine Learning Research*, Vol. 6, pp. 1099–1125.

Micchelli, C. A. and M. Pontil, 2007, "Feature Space Perspectives for Learning the Kernel", *Machine Learning*, Vol. 66, pp. 297–319.

Moguerza, J. M., A. Muñoz, and I. M. de Diego, 2004, "Improving Support Vector Classification via the Combination of Multiple Sources of Information", *Proceedings of the Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops*.

Momma, M. and K. P. Bennett, 2002, "A Pattern Search Method for Model Selection of Support Vector Regression", *Proceedings of the SIAM International Conference on Data Mining*.

Mosek, 2010, *The MOSEK Optimization Tools Manual Version 6.0 (Revision 66)*, MOSEK ApS, Denmark.

Myers, R. H. and D. C. Montgomery, 2002, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Wiley-Interscience.

Nguyen, C. H. and T. B. Ho, 2008, "An Efficient Kernel Matrix Evaluation Measure", *Pattern Recognition*, Vol. 41, No. 11, pp. 3366–3372.

Noble, W. S., 2004, "Support Vector Machine Applications in Computational Biology", Schölkopf, B., K. Tsuda, and J.-P. Vert (eds.), *Kernel Methods in Computational Biology*, chap. 3, The MIT Press.

Ocnlinx, V., V. Wertz, and M. Verleysen, 2009, "Nonlinear Data Projection on Non-Euclidean Manifolds with Controlled Trade-off between Trustworthiness and Continuity", *Neurocomputing*, Vol. 72, pp. 1444–1454.

Ong, C. S. and A. J. Smola, 2003, "Machine Learning using Hyperkernels", *Proceedings of the 20th International Conference on Machine Learning.*

Ong, C. S., A. J. Smola, and R. C. Williamson, 2003, "Hyperkernels", *Advances in Neural Information Processing Systems 15.*

Ong, C. S., A. J. Smola, and R. C. Williamson, 2005, "Learning the Kernel with Hyperkernels", *Journal of Machine Learning Research*, Vol. 6, pp. 1043–1071.

Özen, A., M. Gönen, E. Alpaydın, and T. Haliloğlu, 2009, "Machine Learning Integration for Predicting the Effect of Single Amino Acid Substitutions on Protein Stability", *BMC Structural Biology*, Vol. 9, p. 66.

Pavlidis, P., J. Weston, J. Cai, and W. N. Grundy, 2001, "Gene Functional Classification from Heterogeneous Data", *Proceedings of the 5th Annual International Conference on Computational Molecular Biology.*

Pearson, K., 1901, "On Lines and Planes of Closest Fit to Systems of Points in Space", *Philosophical Magazine*, Vol. 2, No. 6, pp. 559–572.

Pedersen, A. G. and H. Nielsen, 1997, "Neural Network Prediction of Translation Initiation Sites in Eukaryotes: Perspectives for EST and Genome Analysis", *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology.*

Pereira, F. and G. Gordon, 2006, "The Support Vector Decomposition Machine", *Proceedings of the 23rd International Conference on Machine Learning.*

Platt, J. C., N. Cristianini, and J. Shawe-Taylor, 2000, "Large Margin DAGs for Multiclass Classification", *Advances in Neural Information Processing Systems 12.*

Qiu, S. and T. Lane, 2005, "Multiple Kernel Learning for Support Vector Regression", Tech. rep., Computer Science Department, University of New Mexico.

Qiu, S. and T. Lane, 2009, “A Framework for Multiple Kernel Support Vector Regression and Its Applications to siRNA Efficacy Prediction”, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 6, No. 2, pp. 190–199.

Rakotomamonjy, A., F. Bach, S. Canu, and Y. Grandvalet, 2007, “More Efficiency in Multiple Kernel Learning”, *Proceedings of the 24th International Conference on Machine Learning*.

Rakotomamonjy, A., F. R. Bach, S. Canu, and Y. Grandvalet, 2008, “SimpleMKL”, *Journal of Machine Learning Research*, Vol. 9, pp. 2491–2521.

Rifkin, R. and A. Klautau, 2004, “In Defense of One-Vs-All Classication”, *Journal of Machine Learning Research*, Vol. 5, pp. 101–141.

Roweis, S. and L. Saul, 2000, “Nonlinear Dimensionality Reduction by Locally Linear Embedding”, *Science*, Vol. 290, pp. 2323–2326.

Schapire, R. E., 1990, “The Strength of Weak Learnability”, *Machine Learning*, Vol. 5, No. 2, pp. 197–227.

Schmidt, M. and H. Gish, 1996, “Speaker Identification via Support Vector Classifiers”, *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*.

Schölkopf, B. and A. J. Smola, 2002, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, Cambridge, MA.

Schölkopf, B., K. Tsuda, and J.-P. Vert (eds.), 2004a, *Kernel Methods in Computational Biology*, The MIT Press.

Schölkopf, B., K. Tsuda, and J.-P. Vert, 2004b, “A Primer on Kernel Methods”, Schölkopf, B., K. Tsuda, and J.-P. Vert (eds.), *Kernel Methods in Computational Biology*, chap. 2, The MIT Press.

Silverman, B. W., 1985, “Some Aspects of the Spline Smoothing Approach to Non-parametric Regression Curve Fitting”, *Journal of the Royal Statistical Society: Series B*, Vol. 47, pp. 1–52.

Sonnenburg, S., G. Rätsch, and C. Schäfer, 2006a, “A General and Efficient Multiple Kernel Learning Algorithm”, *Advances in Neural Information Processing Systems 18*.

Sonnenburg, S., G. Rätsch, C. Schäfer, and B. Schölkopf, 2006b, “Large Scale Multiple Kernel Learning”, *Journal of Machine Learning Research*, Vol. 7, pp. 1531–1565.

Subrahmanya, N. and Y. C. Shin, 2010, “Sparse Multiple Kernel Learning for Signal Processing Applications”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 5, pp. 788–798.

Sugiyama, M., 2007, “Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis”, *Journal of Machine Learning Research*, Vol. 8, pp. 1027–1061.

Szafranski, M., Y. Grandvalet, and A. Rakotomamonjy, 2008, “Composite Kernel Learning”, *Proceedings of the 25th International Conference on Machine Learning*.

Tan, Y. and J. Wang, 2004, “A Support Vector Machine with a Hybrid Kernel and Minimal Vapnik-Chervonenkis Dimension”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 4, pp. 385–395.

Tanabe, H., T. B. Ho, C. H. Nguyen, and S. Kawasaki, 2008, “Simple but Effective Methods for Combining Kernels in Computational Biology”, *Proceedings of IEEE International Conference on Research, Innovation and Vision for the Future.*

Tao, D., X. Li, W. Hu, S. Maybank, and X. Wu, 2005a, “Supervised Tensor Learning”, *Proceedings of the 5th IEEE International Conference on Data Mining.*

Tao, D., X. Li, X. Wu, and S. J. Maybank, 2009, "Geometric Mean for Subspace Selection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, pp. 260–274.

Tao, Q., G.-W. Wu, F.-Y. Wang, and J. Wang, 2005b, "Posterior Probability Support Vector Machines for Unbalanced Data", *IEEE Transactions on Neural Networks*, Vol. 16, No. 6, pp. 1561–1573.

Tenenbaum, J., V. de Silva, and J. Langford, 2000, "A Global Geometric Framework for Nonlinear Dimensionality Reduction", *Science*, Vol. 290, pp. 2319–2323.

Tipping, M. E. and C. M. Bishop, 1999, "Mixtures of Probabilistic Principal Component Analyzers", *Neural Computation*, Vol. 11, pp. 443–482.

Tsang, I. W.-H. and J. T.-Y. Kwok, 2006, "Efficient Hyperkernel Learning Using Second-Order Cone Programming", *IEEE Transactions on Neural Networks*, Vol. 17, No. 1, pp. 48–58.

Tsuda, K., S. Uda, T. Kin, and K. Asai, 2004, "Minimizing the Cross Validation Error to Mix Kernel Matrices of Heterogeneous Biological Data", *Neural Processing Letters*, Vol. 19, No. 1, pp. 63–72.

Vapnik, V., 1998, *The Nature of Statistical Learning Theory*, John Wiley & Sons.

Varma, M. and B. R. Babu, 2009, "More Generality in Efficient Multiple Kernel Learning", *Proceedings of the 26th International Conference on Machine Learning.*

Varma, M. and D. Ray, 2007, "Learning the Discriminative Power-Invariance Trade-off", *Proceedings of the International Conference in Computer Vision.*

Weinberger, K. Q. and L. K. Saul, 2009, "Distance Metric Learning for Large Margin Nearest Neighbor Classification", *Journal of Machine Learning Research*, Vol. 10, pp. 207–244.

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, 2001, "Feature Selection for SVMs", *Advances in Neural Information Processing Systems 13.*

Weston, J. and C. Watkins, 1998, "Multi-class Support Vector Machines", Tech. Rep. CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science.

Wilcoxon, F., 1945, "Individual Comparisons by Ranking Methods", *Biometrics*, Vol. 1, pp. 80–83.

Wu, M., B. Schölkopf, and G. Bakır, 2006, "A Direct Method for Building Sparse Kernel Learning Algorithms", *Journal of Machine Learning Research*, Vol. 7, pp. 603–624.

Wu, M., K. Yu, S. Yu, and B. Schölkopf, 2007, "Local Learning Projections", *Proceedings of the 24th International Conference on Machine Learning.*

Xu, L., J. Neufeld, B. Larson, and D. Schuurmans, 2005, "Maximum Margin Clustering", *Advances in Neural Processing Systems 17.*

Xu, Z., R. Jin, I. King, and M. R. Lyu, 2009a, "An Extended Level Method for Efficient Multiple Kernel Learning", *Advances in Neural Information Processing Systems 21.*

Xu, Z., R. Jin, J. Ye, M. R. Lyu, and I. King, 2009b, "Non-Monotonic Feature Selection", *Proceedings of the 26th International Conference on Machine Learning.*

Yamanishi, Y., F. Bach, and J.-P. Vert, 2007, "Glycan Classification with Tree Kernels", *Bioinformatics*, Vol. 23, No. 10, pp. 1211–1216.

Yan, F., K. Mikolajczyk, J. Kittler, and M. Tahir, 2009, "A Comparison of l_1 Norm and l_2 Norm Multiple Kernel SVMs in Image and Video Classification", *Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing.*

Yan, S., D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, 2007, "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, pp. 40–51.

Yang, J., Y. Li, Y. Tian, L. Duan, and W. Gao, 2009, "Group-Sensitive Multiple Kernel Learning for Object Categorization", *Proceedings of the 12th IEEE International Conference on Computer Vision*.

Yang, J., Y. Li, Y. Tian, L. Duan, and W. Gao, 2010, "Per-Sample Multiple Kernel Approach for Visual Concept Learning", *EURASIP Journal on Image and Video Processing*.

Ye, J., J. Chen, and S. Ji, 2007a, "Discriminant Kernel and Regularization Parameter Learning via Semidefinite Programming", *Proceedings of the 24th International Conference on Machine Learning*.

Ye, J., S. Ji, and J. Chen, 2007b, "Learning the Kernel Matrix in Discriminant Analysis via Quadratically Constrained Quadratic Programming", *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ye, J., S. Ji, and J. Chen, 2008, "Multi-Class Discriminant Kernel Learning via Convex Programming", *Journal of Machine Learning Research*, Vol. 9, pp. 719–758.

Ying, Y., K. Huang, and C. Campbell, 2009, "Enhanced Protein Fold Recognition through a Novel Data Integration Approach", *BMC Bioinformatics*, Vol. 10, No. 1, p. 267.

Zhao, B., J. T. Kwok, and C. Zhang, 2009, "Multiple Kernel Clustering", *Proceedings of the 9th SIAM International Conference on Data Mining*.

Zien, A. and C. S. Ong, 2007, "Multiclass Multiple Kernel Learning", *Proceedings of the 24th International Conference on Machine Learning*.

Zien, A. and C. S. Ong, 2008, "An Automated Combination of Kernels for Predicting Protein Subcellular Localization", *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics.*