Strategies for Similarity-based Learning

Yihua Chen

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2010

Program Authorized to Offer Degree: Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Yihua Chen

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

Maya R. Gupta

Reading Committee:

_____

Maryam Fazel

_____

Maya R. Gupta

_____

Mari Ostendorf

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Strategies for Similarity-based Learning

Yihua Chen

Chair of the Supervisory Committee:
Professor Maya R. Gupta
Electrical Engineering

This dissertation addresses the problem of learning from similarity data. Such similarity-based learning problems arise in bioinformatics, computer vision, information retrieval, natural language processing, and a broad range of other fields. I first review the field of similarity-based classification. Then I propose two design goals for similarity-based weighted nearest-neighbor classifiers. Based on these design goals, two weighting methods are proposed. Then a comparative study of ten similarity-based classification methods is presented with a comprehensive set of experimental results on eight real data sets.

Similarity measures in many real applications generate indefinite similarity matrices. These indefinite kernels can be problematic for standard kernel-based learning algorithms as the optimization problems become nonconvex and the underlying theory is invalidated. In order to adapt kernel methods for similarity-based learning, I introduce a method that aims to simultaneously find a reproducing kernel Hilbert space based on the given indefinite similarities and train a classifier with good generalization in that space. The method is formulated as a convex optimization problem. I also propose a simplified version that can reduce overfitting and whose associated convex optimization problem can be solved efficiently. The proposed simplified version is extended to multiple kernel learning, where indefinite similarities are combined with multiple kernels for the learning task.

Lastly, I discuss the research on completing a kernel matrix with missing rows and columns using auxiliary information, and sketch a possible direction for future research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Plain language is not enough to express my deep gratitude to those who made this dissertation possible, but to say a few "thanks" is the least I can do here.

First, my thanks go to my supervisory committee: Maryam Fazel, Maya Gupta, Bill Noble, and Mari Ostendorf. I would like to thank Mari for her statistical language processing class, which introduced me to many machine learning problems in natural language processing, and also her suggestions on how to evaluate posterior probability estimates on real data sets. This dissertation uses optimization heavily, almost all of which I learned from Maryam's convex optimization class; the math taught in her class was so delightful that I could only get satisfied by doing a lot of extra homework problems. Unlike some literature on learning with indefinite similarities that only presents experimental results on artificial data sets, all the experiments reported in this dissertation were done on real data sets, and some of them were provided by Bill. I also want to thank Bill for pointing me to many papers related to my research. To have had Maya as my academic adviser is one of the most wonderful things that happened to me during graduate school, and this dissertation serves as a witness to her constant inspiration.

More thanks go to those whom I worked with in the past four years: Hyrum Anderson, Sergey Feldman, Bela Frigyik, Eric Garcia, Evan Hanusa, Nasiha Hrustemovic, Kevin Jamieson, Alex Marin, Will Mortensen, Nathan Parrish, Peter Sadowski, Eric Swanson, and Kristi Tsukida. They simply made my graduate school years a lot more fun.

I also appreciate guidance, help, inspiration, or mentorship from Luca Cazzanti, John Krumm, Kunal Mukerjee, Ting-Kei Pong, Ali Rahimi, Ben Recht, Paul Tseng, and I apologize for any appreciation left unmentioned.

Last but not least, I would like to thank my family, especially my parents and my uncle, for their constant support and encouragement.

DEDICATION

To my parents

with love and gratitude

Chapter 1

INTRODUCTION

Man is a classifying animal: in one sense it may be said that the whole process of speaking is nothing but distributing phenomena, of which no two are alike in every respect, into different classes on the strength of perceived similarities and dissimilarities.

Otto Jespersen

Two common types of data in machine learning problems are vectorial data and pairwise similarity data. Vectorial data represents samples as points in an n-dimensional Euclidean space, while similarity data reveals pairwise (dis)similarities between samples, and is usually presented in the form of a matrix. Many theories and algorithms have been developed to solve learning problems for vectorial data. However, as put by Laub et al. [58], "structure is really what is of interest in data analysis and not a particular representation." Sometimes, the inherent structure of the data can only be properly described by a pairwise relationship, often interpreted as (dis)similarity. Such data occurs natively in computer vision, bioinformatics, information retrieval, natural language processing, and a broad range of other fields, especially those involving or trying to imitate human judgment.

This research focuses on the problem of classification given pairwise similarity data, although the methods considered here are applicable to a range of similarity-based estimation tasks, such as ranking and regression. Formally, similarity-based classifiers estimate the class label of a test sample based on the similarities between the test sample and a set of labeled training samples, and also the pairwise similarities between the training samples. Like others, I use the term *similarity-based classification* whether the pairwise relationship is a similarity or dissimilarity. Similarity-based classification does not require direct access

to the features of the samples, and thus the sample space can be any set, not necessarily a Euclidean space, as long as the similarity function is well defined for any pair of samples. Moreover, for certain types of samples whose features we can access directly, pairwise similarities computed from their features can still be more helpful than their original features for solving the learning problem.

To formally describe the similarity-based classification problem, let Ω be the sample space and $\mathcal{G}$ be the finite set of class labels. Let $\psi : \Omega \times \Omega \to \mathbb{R}$ be the similarity function. I assume that the pairwise similarities between n training samples are given as an $n \times n$ similarity matrix S whose (i, j)-entry is $\psi(x_i, x_j)$, where $x_i \in \Omega$, $i = 1, \ldots, n$, denotes the ith training sample, and $y_i \in \mathcal{G}$, $i = 1, \ldots, n$, the corresponding ith class label. The problem is to estimate the class label $\hat{y}$ for a test sample x based on its similarities to the training samples $\psi(x, x_i)$, $i = 1, \ldots, n$ and its self-similarity $\psi(x, x)$.

Similarity functions can be asymmetric or fail to satisfy the other mathematical properties required for metrics or inner products [85]. Some simple example similarity functions are: travel time from one place to another, compressability of one random process given a code built for another, and the minimum number of steps to convert one sequence into another (edit distance). Computer vision researchers use many similarities, such as the tangent distance [26], earth mover's distance (EMD) [83], shape matching distance [11], and pyramid match kernel [40] to measure the similarity or dissimilarity between images in order to do image retrieval and object recognition. In bioinformatics, the Smith-Waterman algorithm [89], the FASTA algorithm [61], and the BLAST algorithm [2] are popular methods to compute the similarity between different amino acid sequences for protein classification. In natural language processing, researchers use similarity measures such as matching coefficient, Dice coefficient, Jaccard coefficient, and overlap coefficient to estimate the semantic similarity between words or documents in the vector space model [66]. The cosine similarity between term frequency-inverse document frequency (tf-idf) vectors is widely used in information retrieval and text mining for document classification.

Notions of similarity appear to play a fundamental role in human learning, and thus psychologists have done extensive research to model human similarity judgment. Tversky's *contrast model* and *ratio model* [98] represent an important class of similarity functions.

In these two models, each sample is represented by a set of features, and the similarity function is an increasing function of set overlap but a decreasing function of set differences. Tversky's set-theoretic similarity models have been successful in explaining human judgment in various similarity assessment tasks, and are consistent with the observations made by psychologists that metrics do not account for cognitive judgment of similarity in complex situations [98, 99, 33]. Therefore, similarity-based classification may be useful for imitating or understanding how humans categorize.

This dissertation details the contributions I have made to date in the research area of similarity-based learning, which include:

1. design goals and new methods for similarity-based weighted nearest-neighbor classifiers,

2. a comprehensive set of experimental results for eight similarity-based learning problems using ten different similarity-based classification methods,

3. a new method that simultaneously learns a kernel through spectrum modification and trains a discriminative classifier given indefinite similarities, and

4. an extension of the above method to combining indefinite similarities with multiple kernels for classification.

Some of these contributions have previously appeared in the peer-reviewed literature [21, 23, 22].

To begin with, Chapter 2 reviews the background and related work. Then in Chapter 3, I propose design goals for similarity-based weighted nearest-neighbors and also methods that satisfy these design goals. The results of a comparative study of ten similarity-based classification methods are reported and discussed in Chapter 4. In Chapter 5, after introducing a method that aims to simultaneously learn a kernel and train a discriminative classifier given indefinite similarities, I propose a simplified version as a trade-off between model flexibility and overfitting, and also discuss its efficient implementations. At the end

of Chapter 5, I extend the proposed kernel learning method to combining indefinite similarities with multiple kernels. Chapter 6 departs from the contributions I have made and lands on a topic that I am interested in for my future research; the speculative ideas discussed in that chapter serve as a proof that my intellectual curiosity has not diminished at the end of graduate school. Chapter 7 summarizes the whole dissertation.

Chapter 2

BACKGROUND AND RELATED WORK

Tell us what the former things were, so that we may consider them and know their final outcome.

Isaiah 41:22

A natural approach to similarity-based classification is k-nearest-neighbors (k-NN), because k-NN does not require metric properties. Nearest-neighbor learning is the algorithmic parallel of the *exemplar* model of human learning [35]. Although simple, it is often effective, and experiments on some benchmark data sets show that it is difficult to surpass the performance of k-NN for similarity data [75]. Weighted k-NN is a generalization of k-NN by assigning weights to the neighbors, which is detailed in Chapter 3.

A simple alternative to k-NN classifiers for similarity-based classification is the nearest-centroid classifier [102], which is a simple algorithmic parallel of the *prototype* model of human learning [35], and generalizes the nearest-means classifier for Euclidean features. The nearest-centroid classifier models each class by one central prototype, and classifies the test sample as the class whose prototype is most similar to the test sample. A more flexible variant that takes into account class variability is *similarity discriminant analysis* (SDA), which uses an exponential model based on the maximum entropy principle for the similarity from a sample to each class centroid [18]. These model-based approaches may have a high model bias due to the rigidity of using only one prototype per class. In order to reduce the model bias, local versions have been proposed that act on the k-nearest neighbors of the test sample [16, 18]. However, degenerate cases tend to occur quite often when learning a local SDA model. In order to handle this degeneracy problem, three regularization methods were investigated in [17], among which directly regularizing the class-conditional probabilities seems to be most effective. Furthermore, Sadowski et al. proposed a Bayesian framework for local SDA, in which a Dirichlet prior was used to

achieve regularization [84]. They also formulated a variant of local SDA that models pairwise similarities between the test sample and all its neighbors, rather than those between the test sample and local centroids [84].

Other similarity-based classification methods are mainly based on two different perspectives of similarities. One is to treat similarities as inner products, and the other is to treat similarities as features. These two different perspectives are discussed in Section 2.1 and Section 2.2, respectively.

2.1 Similarities as Inner Products

A popular approach to similarity-based classification is to treat the given similarities as inner products in some Hilbert space or to treat dissimilarities as distances in some Euclidean space. This approach can be roughly divided into two categories: one is to explicitly embed the samples in a Euclidean space according to the given (dis)similarities using multidimensional scaling (see [12] for further reading) and then apply classifiers for Euclidean features; the other is to modify the similarities to be kernels and apply kernel classifiers, which is the topic of this section.

2.1.1 Support Vector Machine

Here I focus on the support vector machine (SVM), which is a well-known representative of kernel methods, and thus appears to be a natural approach to similarity-based learning. All the SVM algorithms discussed here are for binary classification (see [48, 80] for multiclass SVM) so that $y_i \in \{\pm 1\}$. An elegant interpretation of the SVM is to view it as a special case of empirical risk minimization (ERM) with regularization, which solves

$$\underset{f \in \mathcal{H}_K}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^{n} L_{\text{hinge}}(y_i, f(x_i)) + \eta \|f\|_K^2 , \tag{2.1}$$

where $\mathcal{H}_K$ is the reproducing kernel Hilbert space (RKHS) induced by a kernel function $K : \Omega \times \Omega \to \mathbb{R}$, $\|f\|_K = \sqrt{\langle f, f \rangle_K}$ is the norm induced by the inner product associated with $\mathcal{H}_K$, $\eta > 0$ is the regularization parameter, and L_{hinge} is the hinge loss defined by

$$L_{\text{hinge}}(y, f(x)) \triangleq \max(1 - yf(x), 0) .$$

The decision rule with the discriminant function $f(x)$ is simply

$$\hat{y} = \operatorname{sgn}(f(x)).$$

By the representer theorem [87], the solution to (2.1) has the form

$$f(x) = \sum_{i=1}^{n} c_i K(x, x_i). \tag{2.2}$$

By adding a bias term b to (2.2), that is, letting

$$f(x) = \sum_{i=1}^{n} c_i K(x, x_i) + b,$$

we can write the primal problem of the SVM as

$$\begin{aligned} \underset{c,\, b,\, \xi}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta c^T K c \\ \text{subject to} \quad & \operatorname{diag}(y)(Kc + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \end{aligned} \tag{2.3}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}$, and $\xi \in \mathbb{R}^n$ are variables, $\mathbf{1}$ denotes the column vector with all entries one, y is the $n \times 1$ vector whose ith element is y_i, K is an $n \times n$ positive semidefinite (PSD) kernel matrix whose (i, j)-entry is $K(x_i, x_j)$, and $\geq$ denotes component-wise inequality for vectors. In practice, people often prefer to solve the dual problem of (2.3), that is,

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \mathbf{1}^T\alpha - \frac{1}{2}\alpha^T \operatorname{diag}(y) K \operatorname{diag}(y)\alpha \\ \text{subject to} \quad & 0 \leq \alpha \leq C\mathbf{1}, \\ & y^T\alpha = 0, \end{aligned} \tag{2.4}$$

with variable $\alpha \in \mathbb{R}^n$ and hyperparameter $C = \frac{1}{2n\eta} > 0$. By solving (2.4), we can recover c by

$$c_i = y_i\alpha_i, \quad i = 1, \ldots, n,$$

and recover b by

$$b = y_j - \sum_{i=1}^{n} c_i K(x_i, x_j)$$

for any j that satisfies $0 < \alpha_j < C$.

The theory of RKHS requires the kernel function to satisfy Mercer's condition, and thus the corresponding kernel matrix K must be PSD, which makes both (2.3) and (2.4) quadratic programs. However, many similarity functions do not satisfy the properties of an inner product, and thus the similarity matrix S can be indefinite. In the following subsections, I will discuss several methods to modify similarities into kernels; a previous review can be found in [104]. Unless mentioned otherwise, in this section I assume that S is symmetric. If not, its symmetric part $\frac{1}{2}\left(S + S^T\right)$ is used instead. Notice that the symmetrization does not affect the objective function in (2.4) since

$$\begin{aligned}\alpha^T \operatorname{diag}(y)\frac{1}{2}(S + S^T)\operatorname{diag}(y)\alpha &= \frac{1}{2}\alpha^T \operatorname{diag}(y)S\operatorname{diag}(y)\alpha + \frac{1}{2}\alpha^T \operatorname{diag}(y)S^T\operatorname{diag}(y)\alpha \\ &= \alpha^T \operatorname{diag}(y)S\operatorname{diag}(y)\alpha.\end{aligned}$$

2.1.2 Indefinite Kernels

One approach is to simply replace K with S, and ignore the fact that S is indefinite. For example, although the SVM dual problem given by (2.4) is no longer convex when S is indefinite, Lin and Lin showed that the sequential minimal optimization (SMO) algorithm [77, 78] will still converge with a simple modification to the original algorithm [60], but the solution is a stationary point instead of a global maximum. Ong et al. interpreted this as finding the stationary point in a reproducing kernel Kreĭn space (RKKS) [73], while Haasdonk showed that this is equivalent to minimizing the distance between reduced convex hulls in a pseudo-Euclidean space [43]. A Kreĭn space, denoted by $\mathcal{K}$, is defined to be the direct sum of two disjoint Hilbert spaces, denoted by $\mathcal{H}_+$ and $\mathcal{H}_-$, respectively. So for any $a, b \in \mathcal{K} = \mathcal{H}_+ \oplus \mathcal{H}_-$, there are unique $a_+, b_+ \in \mathcal{H}_+$ and unique $a_-, b_- \in \mathcal{H}_-$ such that $a = a_+ + a_-$ and $b = b_+ + b_-$. The "inner product" on $\mathcal{K}$ is defined by

$$\langle a, b\rangle_{\mathcal{K}} = \langle a_+, b_+\rangle_{\mathcal{H}_+} - \langle a_-, b_-\rangle_{\mathcal{H}_-},$$

which no longer has the property of positive definiteness. Pseudo-Euclidean space is a special case of Kreĭn space where $\mathcal{H}_+$ and $\mathcal{H}_-$ are two Euclidean spaces. Ong et al. provided a representer theorem for RKKS that poses learning in RKKS as a problem of finding a stationary point of the risk functional [73], in contrast to minimizing a risk functional in

RKHS. Using indefinite kernels in ERM methods such as SVM can lead to a saddle point solution and thus does not ensure minimizing the risk functional, so this approach does not guarantee learning in the sense of a good function approximation. Also, the nonconvexity of the problem may require intensive computation.

2.1.3 Spectrum Clip

Since S is assumed to be symmetric, it has an eigenvalue decomposition $S = U\Lambda U^T$, where U is an orthogonal matrix and Λ is a diagonal matrix of real eigenvalues, that is, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Spectrum clip makes S PSD by clipping all the negative eigenvalues to zero. Some researchers assume that the negative eigenvalues of the similarity matrix are caused by noise and view spectrum clip as a denoising step [104]. Let

$$\Lambda_{\text{clip}} = \text{diag}\left(\max(\lambda_1, 0), \ldots, \max(\lambda_n, 0)\right),$$

and the modified PSD similarity matrix be $S_{\text{clip}} = U\Lambda_{\text{clip}}U^T$. Let v_i denote the ith column vector of U^T. Using S_{clip} as a kernel matrix for training the SVM is equivalent to implicitly using $x_i = \Lambda_{\text{clip}}^{1/2} v_i$ as the representation of the ith training sample since $\langle x_i, x_j \rangle$ is equal to the (i, j)-entry of S_{clip}. A mathematical justification for spectrum clip is that S_{clip} is the nearest PSD matrix to S in terms of the Frobenius norm [44], that is,

$$S_{\text{clip}} = \arg\min_{K \succeq 0} \|K - S\|_F,$$

where $\succeq$ denotes the generalized inequality with respect to the PSD cone.

2.1.4 Spectrum Flip

In contrast to the interpretation that negative eigenvalues are caused by noise, some researchers show that the negative eigenvalues of some similarity data can code useful information about object features or categories [57, 58], which agrees with some fundamental psychological studies [99, 32]. In order to use the negative eigenvalues, Graepel et al. proposed an SVM in pseudo-Euclidean space [38], and Pekalska et al. also considered a generalized nearest mean classifier and Fisher linear discriminant classifier in the same

space [75]. Following the notation in Section 2.1.2, they assume that the samples lie in a Kreĭn space $\mathcal{K} = \mathcal{H}_+ \oplus \mathcal{H}_-$ with similarities given by

$$\psi(a, b) = \langle a_+, b_+ \rangle_{\mathcal{H}_+} - \langle a_-, b_- \rangle_{\mathcal{H}_-} .$$

These proposed classifiers are their standard versions in the Hilbert space $\mathcal{H} = \mathcal{H}_+ \oplus \mathcal{H}_-$ with associated inner product

$$\langle a, b \rangle_{\mathcal{H}} = \langle a_+, b_+ \rangle_{\mathcal{H}_+} + \langle a_-, b_- \rangle_{\mathcal{H}_-} .$$

This is equivalent to flipping the sign of the negative eigenvalues of the similarity matrix S: let

$$\Lambda_{\text{flip}} = \text{diag}\left(|\lambda_1|, \ldots, |\lambda_n|\right),$$

and the similarity matrix after spectrum flip is $S_{\text{flip}} = U\Lambda_{\text{flip}}U^T$. Wu et al. note that this is the same as replacing the original eigenvalues of S with its singular values [104].

2.1.5 Spectrum Shift

Spectrum shift is another popular approach to modifying a similarity matrix into a kernel matrix: since $S + \lambda I = U(\Lambda + \lambda I)U^T$, where I is the identity matrix, any indefinite similarity matrix can be made PSD by shifting its spectrum by the absolute value of its minimum eigenvalue $|\lambda_{\min}(S)|$. Let

$$\Lambda_{\text{shift}} = \Lambda + |\min\left(\lambda_{\min}(S), 0\right)|I,$$

which is used to form the modified similarity matrix $S_{\text{shift}} = U\Lambda_{\text{shift}}U^T$. Compared with spectrum clip and flip, spectrum shift only enhances all the self-similarities by the amount of $|\lambda_{\min}(S)|$ and does not change the similarity between any two different samples. Roth et al. proposed spectrum shift for clustering nonmetric proximity data; they showed that S_{shift} preserves the group structure of the original data represented by S [82]. Let $\mathcal{X}$ be the set of samples to cluster, and $\{\mathcal{X}_\ell\}_{\ell=1}^N$ be a partition of $\mathcal{X}$ into N sets. Specifically, they

consider minimizing the clustering cost function[1]

$$f\left(\{\mathcal{X}_\ell\}_{\ell=1}^N\right) = -\sum_{\ell=1}^{N} \sum_{\substack{i,j\in\mathcal{X}_\ell \\ i\neq j}} \frac{\psi(x_i,x_j)}{|\mathcal{X}_\ell|}, \tag{2.5}$$

where $|\mathcal{X}_\ell|$ denotes the cardinality of set $\mathcal{X}_\ell$. It is easy to see that (2.5) is invariant under spectrum shift.

Recently, Zhang et al. proposed training an SVM only on the k-nearest neighbors of each test sample, called SVM-KNN [108]. They used spectrum shift to produce a kernel from the similarity data. Their experimental results on image classification demonstrate that SVM-KNN performs comparably to a standard SVM classifier, yet is able to trade an increase in test time for a significant reduction in training time.

2.1.6 Spectrum Square

The fact that $SS^T \succeq 0$ for any $S \in \mathbb{R}^{n\times n}$ led us to consider using SS^T as a kernel, which is valid even when S is not symmetric. For symmetric S, this is equivalent to squaring its spectrum since $SS^T = U\Lambda^2U^T$. It is also true that using SS^T is the same as defining a new similarity function $\tilde{\psi}$ for any $a, b \in \Omega$ as

$$\tilde{\psi}(a,b) = \sum_{i=1}^{n} \psi(a,x_i)\psi(x_i,b).$$

Note that for symmetric S, treating SS^T as a kernel matrix K is equivalent to representing each x_i by its similarity feature vector

$$s_i = \begin{bmatrix}\psi(x_i,x_1) & \dots & \psi(x_i,x_n)\end{bmatrix}^T,$$

since $K_{ij} = \langle s_i, s_j\rangle$. The concept of treating similarities as features is discussed in more detail in Section 2.2.

2.2 Similarities as Features

Similarity-based classification problems can be formulated into standard learning problems in Euclidean space by treating the similarities between a sample x and the n training

[1]They originally used dissimilarities in their cost function, and we reformulate it into similarities with the assumption that the relationship between dissimilarities and similarities is affine.

samples as features [38, 39, 75, 74, 59], that is, representing sample x by its similarity feature vector s:

$$s = \begin{bmatrix} \psi(x, x_1) & \dots & \psi(x, x_n) \end{bmatrix}^T.$$

As detailed in Section 2.3, the generalization analysis yields different results for using similarities as features and using similarities as inner products.

Graepel et al. considered applying a linear SVM on similarity feature vectors by solving the following problem [38]:

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} L_{\text{hinge}}(y_i, w^T s_i + b) \tag{2.6}$$

with variables $w \in \mathbb{R}^n$, $b \in \mathbb{R}$ and hyperparameter $C > 0$. Liao and Noble also proposed to apply an SVM on similarity feature vectors [59]; they used a Gaussian radial basis function (RBF) kernel.

In order to make the solution w sparser, which helps ease the computation of the discriminant function $f(s) = w^T s + b$, Graepel et al. substituted the ℓ_1-norm regularization for the squared ℓ_2-norm regularization in (2.6), and proposed a linear programming (LP) machine [39]:

$$\underset{w,b}{\text{minimize}} \quad \|w\|_1 + C\sum_{i=1}^{n} L_{\text{hinge}}(y_i, w^T s_i + b). \tag{2.7}$$

Balcan et al. provided a theoretical analysis for using similarities as features [8], and showed that if a similarity is good in the sense that the expected intraclass similarity is sufficiently large compared to the expected interclass similarity, then given n training samples, there exists a linear separator on the similarities as features that has a specifiable maximum error at a margin that depends on n. Specifically, Theorem 4 in [8] gives a sufficient condition on the similarity function ψ for (2.6) to achieve good generalization. Their latest results for ℓ_1-margin (inversely proportional to $\|w\|_1$) provided similar theoretical guarantees for (2.7) [7, Theorem 11]. Wang et al. showed that under slightly less restrictive assumptions on the similarity function there exists with high probability a convex combination of simple classifiers on the similarities as features which has a maximum specifiable error [101].

Another approach is the potential support vector machine (P-SVM) [45, 52], which solves

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2}\|y - S\alpha\|_2^2 + \epsilon\|\alpha\|_1 \\ \text{subject to} \quad & \|\alpha\|_\infty \leq C, \end{aligned} \tag{2.8}$$

where $C > 0$ and $\epsilon > 0$ are two hyperparameters. Note that by strong duality, (2.8) is equivalent to

$$\underset{\alpha}{\text{minimize}} \quad \frac{1}{2}\|y - S\alpha\|_2^2 + \epsilon\|\alpha\|_1 + \gamma\|\alpha\|_\infty \tag{2.9}$$

for some $\gamma > 0$. One can see from (2.9) that P-SVM is equivalent to the lasso regression [95] with an extra ℓ_∞-norm regularization term. The use of multiple regularization terms in P-SVM is similar to the elastic net [112], which uses ℓ_1 and squared ℓ_2 regularization together.

The algorithms above minimize the empirical risk with regularization. In addition, Pekalska et al. considered generative classifiers for similarity feature vectors; they proposed a regularized Fisher linear discriminant classifier [75] and a regularized quadratic discriminant classifier [74].

We note that treating similarities as features may not capture discriminative information if there is a large intraclass variance compared to the interclass variance, even if the classes are well-separated. A simple example is if the two classes are generated by Gaussian distributions with highly-ellipsoidal covariances, and the similarity function is taken to be a negative linear function of the distance.

2.3 Generalization Bounds for Similarity SVM Classifiers

The analysis of the generalization bounds presented in this section offers a glimpse into the theoretical difference between an SVM using similarity as a kernel and a linear SVM treating similarities as features. In both cases, the SVM learns a linear discriminant function on the similarities:

$$f(s) = w^T s + b,$$

and as mentioned in Section 2.1.1, such a function is learned by minimizing the empirical risk

$$\hat{R}_{\mathcal{D}}(f, L_{\text{hinge}}) = \frac{1}{n}\sum_{i=1}^{n} L_{\text{hinge}}(y_i, f(s_i)),$$

where $\mathcal{D}$ denotes the training set $\{(x_i, y_i)\}_{i=1}^n$, subject to some smoothness constraint ϑ on the function f, that is, by solving

$$\underset{f}{\text{minimize}} \quad \hat{R}_{\mathcal{D}}(f, L_{\text{hinge}}) + \eta_n \vartheta(f),$$

where $\eta_n = \frac{1}{2nC}$. Note that the linear SVM given by (2.6) using (arbitrary) similarities as features corresponds to setting $\vartheta(f) = \|w\|_2^2 = w^T w$, while using (PSD) similarities as a kernel changes the smoothness constraint to $\vartheta(f) = w^T S w$ [81, Appendix B]. In fact, this change of regularizer is the only difference between these two similarity-based SVM approaches; however, as shown below, this seemingly small change in regularization affects the generalization ability of the SVM classifiers.

In order to simplify the analysis, instead of investigating the SVM as presented, we, as standard in SVM learning theory, choose to analyze a slightly modified version of the problem:

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & \hat{R}_{\mathcal{D}}(f, L_{\text{trunc}}) \\ \text{subject to} \quad & \vartheta(f) \leq \beta^2, \end{aligned} \tag{2.10}$$

where the discriminant function f is stripped of the bias term b such that $f(s) = w^T s$, and the loss function is changed to the following truncated hinge loss:

$$L_{\text{trunc}}(y, f(s)) \triangleq \min(L_{\text{hinge}}(y, f(s)), 1) \in [0, 1].$$

The following generalization bound for the SVM using (PSD) similarities[2] as a kernel follows directly from the results in [9].

Theorem 2.1 (Generalization Bound for SVM using Similarities as Kernel)**.** *Suppose (x, y) and $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ are drawn i.i.d. from a distribution on $\Omega \times \{\pm 1\}$. Let ψ be a positive definite similarity function, and there exists $\kappa > 0$ such that $\psi(a, a) \leq \kappa^2$ for all $a \in \Omega$. Let S be the $n \times n$ matrix with (i, j)-entry $\psi(x_i, x_j)$ and s be the $n \times 1$ vector with ith element $\psi(x, x_i)$. For a finite $\beta > 0$, define $\mathcal{F}_{S,\beta}$ as the set of real-valued functions $\left\{ f(s) = w^T s \,\middle|\, w^T S w \leq \beta^2 \right\}$. Then with probability at least $1 - \delta$ with respect to $\mathcal{D}$, every function f in $\mathcal{F}_{S,\beta}$ satisfies*

$$P(yf(s) \leq 0) \leq \hat{R}_{\mathcal{D}}(f, L_{\text{trunc}}) + 4\beta\kappa\sqrt{\frac{1}{n}} + \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

[2] If the original similarities are not PSD, then they must be modified to be PSD before this result applies. A discussion of some common modifications can be found in Section 2.1.

The detailed proof of Theorem 2.1 is provided in [21, Appendix A] by my co-author Eric Garcia.

Theorem 2.1 says that with high probability, as $n \to \infty$, the misclassification rate is tightly bounded by the empirical risk $\hat{R}_{\mathcal{D}}(f, L_{\text{trunc}})$, implying that a discriminant function trained by (2.10) with $\vartheta(f) = w^T S w$ generalizes well to unseen data.

Next, we state a weaker result in Theorem 2.2 for a linear SVM using (arbitrary) similarities as features. First, we randomly select m ($< n$) samples $\tilde{\mathcal{D}} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m$ from the training set as prototypes. Then we let the features of x be its similarities to these m prototypes, represented by an $m \times 1$ vector $\tilde{s}$ with ith element $\psi(x, \tilde{x}_i)$. The following result is obtained on the remaining $n - m$ training samples, denoted by $\bar{\mathcal{D}} = \mathcal{D} \backslash \tilde{\mathcal{D}}$.

Theorem 2.2 (Generalization Bound for SVM using Similarities as Features)**.** *Suppose* (x, y) *and* $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ *are drawn i.i.d. from a distribution on* $\Omega \times \{\pm 1\}$*. Let* ψ *be a similarity function, and there exists* $\kappa > 0$ *such that* $\psi(a, b) \leq \kappa^2$ *for all* $a, b \in \Omega$*. Let* $\tilde{\mathcal{D}} = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m \subset \mathcal{D}$ *be a set of randomly chosen prototypes with* $m < n$*, and denote* $\bar{\mathcal{D}} = \mathcal{D} \backslash \tilde{\mathcal{D}}$*. Let* $\tilde{s}$ *be the* $m \times 1$ *vector with ith element* $\psi(x, \tilde{x}_i)$*, and for a finite* $\beta > 0$*, define* $\mathcal{F}_\beta$ *as the set of real-valued functions* $\left\{ f(\tilde{s}) = w^T \tilde{s} \;\middle|\; w^T w \leq \beta^2 \right\}$*. Then with probability at least* $1 - \delta$ *with respect to* $\bar{\mathcal{D}}$*, every function* f *in* $\mathcal{F}_\beta$ *satisfies*

$$P(yf(\tilde{s}) \leq 0) \leq \hat{R}_{\bar{\mathcal{D}}}(f, L_{\text{trunc}}) + 4\beta\kappa^2 \sqrt{\frac{m}{n}} + \sqrt{\frac{\ln(2/\delta)}{2n}}.$$

The detailed proof of Theorem 2.2 is also provided in [21, Appendix A] by Eric Garcia.

Theorem 2.2 only differs significantly[3] from Theorem 2.1 in the term $\sqrt{m/n}$, which means that if m, the number of prototypes used, grows no faster than $o(n)$, then with high probability, as $n \to \infty$, the misclassification rate is tightly bounded by the empirical risk on the remaining training set $\hat{R}_{\bar{\mathcal{D}}}(f, L_{\text{trunc}})$. This implies that good generalization can achieved by training a linear SVM using similarities to a randomly selected small subset of training samples. However, Theorem 2.2 is unable to claim anything about the generalization when $m = n$, that is, when the entire training set is chosen as prototypes, and this remains as an interesting case for theoretical investigation.

[3]We do not consider having κ in Theorem 2.1 and κ^2 in Theorem 2.2 as a significant difference.

Chapter 3

SIMILARITY-BASED WEIGHTED NEAREST-NEIGHBORS

Be careful the environment you choose for it will shape you; be careful the friends you choose for you will become like them.

W. Clement Stone

In this chapter, I address design goals and propose solutions for weighted nearest-neighbors for similarity-based classification. Most of the research in this chapter also appears in one of my publications [21].

Weighted nearest-neighbor algorithms generalize the simple k-NN algorithm by assigning weights to the neighbors and making each neighbor's voting right proportional to its weight. Weighted k-NN algorithms are task-flexible in the sense that the weights on the neighbors can be interpreted as probabilities as long as they are nonnegative and sum to one. For classification, such weights can be summed for each class to form posterior probabilities, which is helpful for use with asymmetric misclassification costs or when the similarity-based classifier is a component of a larger decision-making system whose other components prefer probabilities. As a lazy learning method, weighted k-NN classifiers do not require training before the arrival of test samples. This might preclude its use in applications with very limited resources for testing, but this can be advantageous to certain applications where the amount of training data is huge, or there are a large number of classes, or the training data is constantly evolving.

3.1 Design Goals for Similarity-based Weighted k-NN

In this chapter, for a test sample x, we use x_i to denote its ith nearest neighbor from the training set as defined by the similarity function ψ for $i = 1, \ldots, k$, and y_i to denote the label of x_i. Also, we redefine S as the $k \times k$ matrix of the similarities between the k-nearest neighbors and s the $k \times 1$ vector of the similarities between the test sample x and its k-

nearest neighbors. For each test sample, weighted k-NN assigns weight w_i to the ith nearest neighbor x_i for $i = 1, \ldots, k$. Weighted k-NN classifies the test sample x as the class $\hat{y}$ that is assigned the most weight:

$$\hat{y} = \arg\max_{g \in \mathcal{G}} \sum_{i=1}^{k} w_i I_{\{y_i = g\}}, \tag{3.1}$$

where $I_{\{\cdot\}}$ is the indicator function. It is common to additionally require that the weights be nonnegative and normalized such that the weights form a posterior distribution over the set of classes $\mathcal{G}$. Then the estimated posterior probability for class g is

$$\hat{P}(Y = g \mid x) = \sum_{i=1}^{k} w_i I_{\{y_i = g\}}, \tag{3.2}$$

which can be used with asymmetric misclassification costs.

Given pairwise similarities, an intuitive and standard approach to weighting nearest neighbors is to give larger weight to neighbors that are more similar to the test sample. Formally, this can be stated as:

Design Goal 1 (Affinity): w_i should be an increasing function of $\psi(x, x_i)$.

In addition, I propose a second design goal for the weights. This design goal is inspired by the observation that in practice, some samples in the training set are often very similar. For example, a random sampling of the emails by one person may include many emails from the same thread that contain repeated text due to replies and forwarding. Another example is to create image training sets for object recognition or other image classification tasks by retrieving images from the Internet; such training sets very often contain images with the same content but different resolutions, formats, compression ratios, or file names [103]. These similar training samples provide highly correlated information to the classifier. In fact, many of the nearest neighbors may provide very similar or even redundant information which can bias the classifier. Moreover, those training samples that are similar to many other training samples should be considered less valuable based on the same motivation for tf-idf. To address this problem, one can choose weights to down-weight highly similar samples and ensure that a diverse set of the neighbors has a voice in the classification decision. This design goal can be formalized as:

Design Goal 2 (Diversity): w_i should be a decreasing function of $\psi(x_i, x_j)$ for any $j \neq i$.

In the next section, I will propose two approaches to weighting neighbors for similarity-based classification that aim to satisfy these two goals.

3.2 Kernel Ridge Interpolation Weights

First, I describe kernel regularized linear interpolation, and I show that it leads to weights that satisfy the two design goals proposed in the previous section. Gupta et al. proposed weights for k-NN in Euclidean space that satisfy a linear interpolation with maximum entropy (LIME) objective [42]:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \left\| \sum_{i=1}^{k} w_i x_i - x \right\|_2^2 - \lambda H(w) \\ \text{subject to} \quad & w_i \geq 0,\ i = 1, \ldots, k, \\ & \sum_{i=1}^{k} w_i = 1, \end{aligned} \tag{3.3}$$

with variable $w \in \mathbb{R}^k$, where $H(w) = -\sum_{i=1}^{k} w_i \log w_i$ is the entropy of the weights and $\lambda > 0$ is a regularization parameter. The first term of the convex objective in (3.3) tries to solve the linear interpolation equations, which balances the weights so that the test point is best approximated by a convex combination of its k-nearest neighbors. Additionally, the entropy maximization pushes the LIME weights toward the uniform weights.

Note that (3.3) can be simplified to a quadratic program (QP) by replacing the negative entropy regularization with a ridge regularizer. Due to the constraint $\mathbf{1}^T w = 1$, the ridge regularizer actually regularizes the variance of the weights and hence has similar effect as the negative entropy regularizer, that is, to push the weights to be uniform. We can write this QP in the following matrix form:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2} w^T X^T X w - x^T X w + \frac{\lambda}{2} w^T w \\ \text{subject to} \quad & w \geq 0, \quad \mathbf{1}^T w = 1, \end{aligned} \tag{3.4}$$

where $X = \begin{bmatrix} x_1 & \cdots & x_k \end{bmatrix}$. Now note that (3.4) is completely specified in terms of the inner products of the feature vectors: $\langle x_i, x_j \rangle$ and $\langle x, x_i \rangle$, and thus we term the solution to (3.4) as *kernel ridge interpolation* (KRI) weights. Generalizing from inner products to similarities,

we form the similarity-based KRI weights:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2} w^T S w - s^T w + \frac{\lambda}{2} w^T w \\ \text{subject to} \quad & w \geq 0, \quad \mathbf{1}^T w = 1. \end{aligned} \tag{3.5}$$

There are three terms in the objective function of (3.5). Acting alone, the linear term $-s^T w$ would give all the weight to the 1-nearest neighbor. This, however, is prevented by the ridge regularization term $\frac{1}{2}\lambda w^T w$, which regularizes the variance of w and hence pushes the weights toward the uniform weights. These two terms work together to give more weight to the training samples that are more similar to the test sample, and therefore help the resulting weights satisfy the first design goal of rewarding neighbors with high affinity to the test sample. The quadratic term in (3.5) can be expanded as follows,

$$\frac{1}{2} w^T S w = \frac{1}{2} \sum_{i,j} \psi(x_i, x_j) w_i w_j.$$

Intuitively, one can see from the above expansion that holding all else constant in (3.5), the bigger $\psi(x_i, x_j)$ and $\psi(x_j, x_i)$ are, the smaller the chosen w_i and w_j will be. Thus the quadratic term tends to down-weight the neighbors that are similar to each other and acts to achieve the second design goal of spreading the weight among a diverse set of neighbors.

A sensitivity analysis further verifies the above observations. Let $g(w; S, s)$ be the objective function of (3.5), and $w^\star$ denote the optimal solution. To simplify the analysis, we assume that $w^\star$ is in the interior of the standard $(k-1)$-simplex and thus

$$\nabla g(w^\star; S, s) = (S + \lambda I) w^\star - s = 0.$$

First, we perturb s by adding $\delta > 0$ to its ith element, that is, we let

$$\tilde{s} = s + \delta e_i,$$

where e_i denotes the standard basis vector with ith element 1 and 0 elsewhere. Then the gradient of g at $w^\star$ with the perturbed $\tilde{s}$ is

$$\nabla g(w^\star; S, \tilde{s}) = (S + \lambda I) w^\star - \tilde{s} = -\delta e_i,$$

and its projection on the standard $(k-1)$-simplex is

$$\nabla g(w^\star; S, \tilde{s}) - \frac{1}{k}\left(\mathbf{1}^T \nabla g(w^\star; S, \tilde{s})\right)\mathbf{1} = \delta\left(\frac{1}{k}\mathbf{1} - e_i\right). \tag{3.6}$$

The direction of the steepest descent given by the negative of the projected gradient in (3.6) indicates that the new optimal solution will have an increased w_i, which satisfies the first design goal.

Next, if we keep s unaltered and perturb S by adding $\delta > 0$ to its (i, j)-entry ($i \neq j$), that is, we let

$$\tilde{S} = S + \delta E_{ij},$$

where E_{ij} denotes the matrix with (i, j)-entry 1 and 0 elsewhere, then the gradient of g at $w^\star$ with the perturbed $\tilde{S}$ is

$$\nabla g(w^\star; \tilde{S}, s) = (\tilde{S} + \lambda I)w^\star - s = \delta E_{ij} w^\star = \delta w_j^\star e_i,$$

and its projection on the standard $(k-1)$-simplex is

$$\nabla g(w^\star; \tilde{S}, s) - \frac{1}{k}\left(\mathbf{1}^T \nabla g(w^\star; \tilde{S}, s)\right)\mathbf{1} = \delta w_j^\star\left(e_i - \frac{1}{k}\mathbf{1}\right). \tag{3.7}$$

The direction of the steepest descent given by the negative of the projected gradient in (3.7) indicates that the new optimal solution will have a decreased w_i, which satisfies the second design goal.

Experimentally, we found little statistically significant difference between using negative entropy or ridge regularization for the KRI weights. Analytically, entropy regularization leads to an exponential form for the weights that can be used to prove consistency [30]. Computationally, the ridge regularizer is more practical because when the S matrix is PSD or approximated by a PSD matrix as discussed in Section 2.1, using ridge regularization results in a QP with box constraints and an equality constraint, which is exactly the same type of QP as the SVM dual problem given by (2.4), and therefore can be solved by the SMO algorithm [77, 78], which has proved to be an efficient algorithm for SVM.

3.2.1 Kernel Ridge Regression Weights

A closed-form solution to (3.5) is possible if one relaxes the problem by removing the constraints $w \geq 0$ and $\mathbf{1}^T w = 1$ which ensure a valid probability distribution formed by the

weights. Then for a PSD S, the objective function $\frac{1}{2}w^T S w - s^T w + \frac{1}{2}\lambda w^T w$ alone is solved by

$$w = (S + \lambda I)^{-1} s. \tag{3.8}$$

Using the above weights, the weighted k-NN decision rule given by (3.1) turns out to be equivalent to classifying by maximizing the discriminant of a local kernel ridge regression. For each class $g \in \mathcal{G}$, local kernel ridge regression (without intercept) solves

$$\underset{\beta_g}{\text{minimize}} \quad \sum_{i=1}^{k} \left(I_{\{y_i = g\}} - \langle \beta_g, \phi(x_i) \rangle \right)^2 + \lambda \langle \beta_g, \beta_g \rangle, \tag{3.9}$$

where ϕ denotes the mapping from the sample space Ω to a Hilbert space with inner product $\langle \phi(x_i), \phi(x_j) \rangle = \psi(x_i, x_j)$. The solution to (3.9) yields the following discriminant function for class g [24]:

$$f_g(x) = \langle \beta_g, \phi(x) \rangle = \nu_g^T (S + \lambda I)^{-1} s,$$

where

$$\nu_g = \begin{bmatrix} I_{\{y_1 = g\}} & \dots & I_{\{y_k = g\}} \end{bmatrix}^T.$$

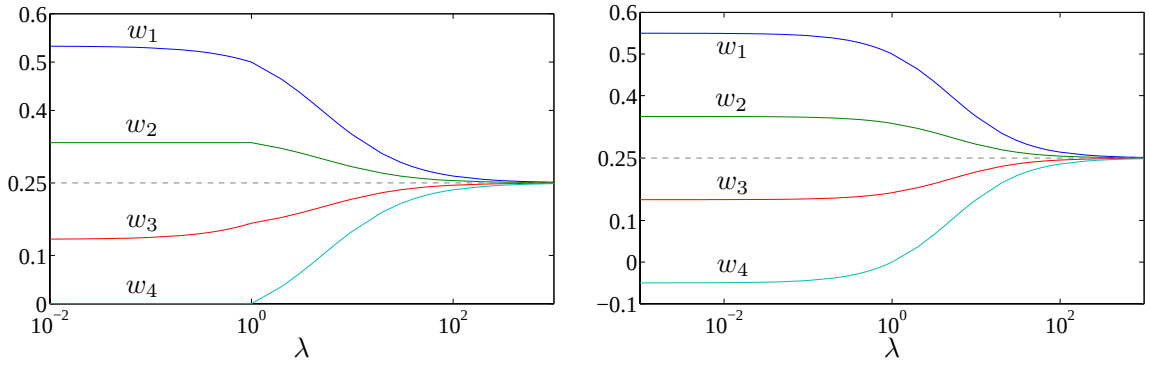
Maximizing $f_g(x)$ over $g \in \mathcal{G}$ produces the same estimated class label $\hat{y}$ as (3.1) using the weights given by (3.8), and hence we refer to these weights as *kernel ridge regression* (KRR) weights.

For an indefinite S, it is possible that $S + \lambda I$ is singular. In the experiments shown in Chapter 4, I will compare handling indefinite S by spectrum clip, flip, shift, or taking the pseudoinverse (pinv), that is,

$$w = (S + \lambda I)^{\dagger} s.$$

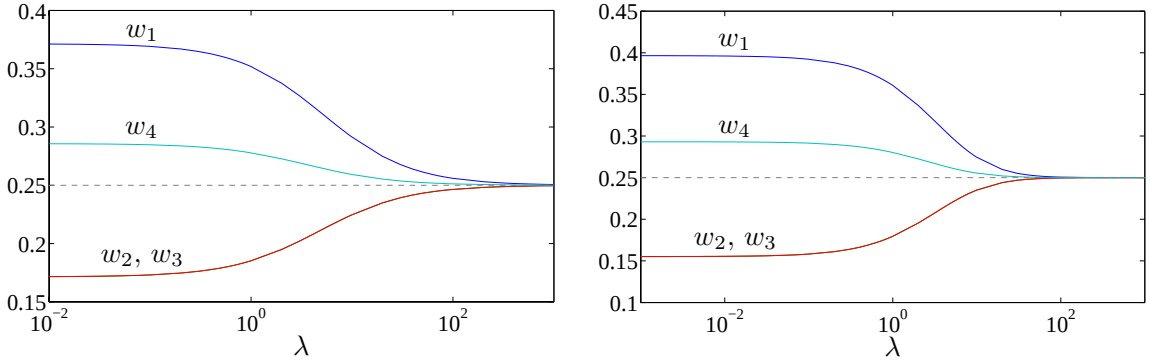
3.2.2 Illustrative Examples

In this subsection, I illustrate the KRI and KRR weights with three toy examples. In each example, there are $k = 4$ nearest-neighbors, and the KRI and KRR weights are shown for a continuous range of regularization parameter λ. For the purpose of comparison, the normalized KRR weights $\tilde{w}$ are shown here where $\tilde{w} = \left(I - \frac{1}{k}\mathbf{1}\mathbf{1}^T\right) w + \frac{1}{k}\mathbf{1}$. This normalization does not affect the result of classification since each weight is shifted by the same constant.



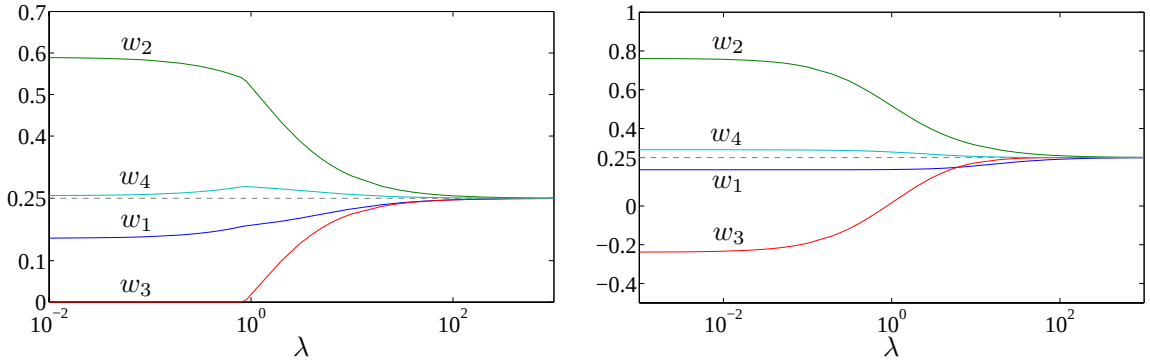
(a) KRI weights (b) KRR weights

Figure 3.1: KRI and KRR weights for Example 1.



(a) KRI weights (b) KRR weights

Figure 3.2: KRI and KRR weights for Example 2.



(a) KRI weights (b) KRR weights

Figure 3.3: KRI and KRR weights for Example 3.

Example 1: $s^T = \begin{bmatrix} 4 & 3 & 2 & 1 \end{bmatrix}$, $S = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$

In Example 1, the design goal of affinity is illustrated. One can see from s that the four distinct training samples are not equally similar to the test sample, and from S that the training samples have zero similarity to each other. As a result, both KRI and KRR give more weight to the training samples that are more similar to the test sample, illustrating that the proposed weighting methods achieve the design goal of affinity. Also, as λ goes larger, the harder the regularization is and the more uniform the weights are.

Example 2: $s^T = \begin{bmatrix} 3 & 3 & 3 & 3 \end{bmatrix}$, $S = \begin{bmatrix} 5 & 1 & 1 & 1 \\ 1 & 5 & 4 & 2 \\ 1 & 4 & 5 & 2 \\ 1 & 2 & 2 & 5 \end{bmatrix}$

In Example 2, the design goal of diversity is illustrated. Here the four training samples are all equally similar to the test sample, but x_2 and x_3 are very similar to each other. As a result, x_2 and x_3 are both weighted down as prescribed by the design goal of diversity. Because of the symmetry of the similarities shown in S, the weights for x_2 and x_3 are exactly the same for both KRI and KRR.

Example 3: $s^T = \begin{bmatrix} 2 & 4 & 3 & 3 \end{bmatrix}$, $S = \begin{bmatrix} 5 & 1 & 1 & 1 \\ 1 & 5 & 4 & 2 \\ 1 & 4 & 5 & 2 \\ 1 & 2 & 2 & 5 \end{bmatrix}$

In Example 3, the interaction between the two design goals is illustrated. The matrix S is the same as in Example 2, but s is no longer uniform. In fact, although x_1 is less similar to the test sample than x_3, x_1 receives more weight than x_3 because it is less similar to the other training samples. The affinity goal allots the largest weight to the most similar neighbor x_2, but because x_2 and x_3 are highly similar to each other, the diversity goal forces them to share weight, resulting in a very small weight assigned to x_3.

Also, one can observe from the juxtaposition of the figures that the KRR weights tend to be smoother than the KRI weights with respect to the change of the regularization parameter λ. This is because the KRI weights are constrained to the standard $(k-1)$-simplex while the KRR weights are unconstrained.

Chapter 4

COMPARATIVE STUDY OF SIMILARITY-BASED CLASSIFICATION METHODS

When they measure themselves by themselves and compare themselves with themselves, they are not wise.

2 Corinthians 10:12

This chapter describes a comparative study of ten similarity-based classification methods: a linear SVM and an RBF SVM using similarities as features, the P-SVM [45, 52], a local SVM (SVM-KNN) [108] and a global SVM using the given similarities as a kernel, local SDA [16, 18], k-NN, and three weighted k-NN methods: the KRI and KRR weights proposed in Chapter 3, and the affinity weights as a control, defined by $w_i = a\psi(x, x_i)$, $i = 1, \ldots, k$, where a is a normalization constant. Note that unlike the proposed KRI and KRR weights, the affinity weights only satisfy the design goal of affinity. Most of the results in this chapter also appear in one of my publications [21].

In addition to the comparative study on classification performance, at the end of this chapter, we also compare weighted k-NN methods for estimating the posterior probabilities of the classes using the uniform, affinity and KRI weights.

4.1 Consistent Treatment of Training and Test Samples

Before we proceed to the experiments, we need to address an important technicality about how to treat training and test samples consistently. Consider the approach that treats similarities as inner products as discussed in Section 2.1. When the given similarity is indefinite, we can perform various spectrum modifications to make an indefinite similarity matrix S into a PSD matrix $\tilde{S}$. Now suppose a test sample x is the same as a training sample x_i; if one trains an ERM classifier such as an SVM using the modified similarities $\tilde{S}$ but uses

the unmodified test similarities, represented by vector

$$s = \begin{bmatrix} \psi(x, x_1) & \dots & \psi(x, x_n) \end{bmatrix}^T,$$

for testing, then the same sample will be treated inconsistently, violating the spirit of ERM. In general, one would like to modify the training and test similarities in a consistent fashion, that is, to modify the underlying similarity function rather than only modifying the similarity matrix S of the training data. In this context, given S and $\tilde{S}$, a transformation T on test similarities is termed *consistent* if $T(s_i)$ is equal to the ith column of $\tilde{S}$ for $i = 1, \dots, n$.

One solution is to modify the training and test samples all at once. However, when test samples are not known beforehand, this may not be possible. For such cases, Wu et al. proposed to first modify S and train the classifier using the modified $n \times n$ similarity matrix $\tilde{S}$, and then for each test sample modify its s in an effort to be consistent with the modified similarities used to train the model [104]. Given s and the self-similarity of the test sample $\psi(x, x)$, their approach is to perform the same spectrum modification on the augmented $(n+1) \times (n+1)$ similarity matrix:

$$S' = \begin{bmatrix} S & s \\ s^T & \psi(x, x) \end{bmatrix}$$

to form $\tilde{S}'$, and then let the modified test similarities $\tilde{s}$ be the first n elements of the last column of $\tilde{S}'$.[1] The classifier trained on the modified training similarities $\tilde{S}$ is then applied on $\tilde{s}$ to obtain the estimated class label $\hat{y}$ for the test sample. To implement this approach, they proposed a fast algorithm to perform eigenvalue decomposition on S' by using the results of the eigenvalue decomposition of S [104]. However, this approach does not guarantee consistency.

To attain consistency, I note that both the spectrum clip and flip modifications can be represented by matrix multiplications, that is, $\tilde{S} = PS$, where P is the corresponding transformation matrix. Therefore, I propose to apply the same transformation matrix P on s such that $\tilde{s} = Ps$, and it is easy to see that according to our previous definition, the transformation $T(s) = Ps$ is consistent. Recall that S has eigenvalue decomposition $S = U\Lambda U^T$,

[1] Note that even if the same spectrum modification is applied to S and S', there is no guarantee that $\tilde{S}$ will be equal to the upper left $n \times n$ submatrix of $\tilde{S}'$.

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. For spectrum flip, the transformation matrix is

$$P_{\text{flip}} = UD_{\text{flip}}U^T,$$

where

$$D_{\text{flip}} = \text{diag}(\text{sgn}(\lambda_1), \ldots, \text{sgn}(\lambda_n)).$$

For spectrum clip, the transformation matrix is

$$P_{\text{clip}} = UD_{\text{clip}}U^T,$$

where

$$D_{\text{clip}} = \text{diag}(I_{\{\lambda_1 \geq 0\}}, \ldots, I_{\{\lambda_n \geq 0\}}),$$

and $I_{\{\cdot\}}$ is the indicator function.

Recall that using the modified similarity matrix $\tilde{S}$ implies embedding the training samples in a Euclidean space. For spectrum clip, the transformation $T(s) = P_{\text{clip}}s$ is equivalent to embedding the test sample x as a feature vector into the same Euclidean space of the embedded training samples:

Proposition 4.1. *Let S_{clip} be the Gram matrix of the column vectors of a matrix $X \in \mathbb{R}^{m \times n}$ with $\text{rank}(X) = m \leq n$. For a given vector $s \in \mathbb{R}^n$, let $x = \arg\min_{z \in \mathbb{R}^m} \left\| X^T z - s \right\|_2$, then $X^T x = P_{\text{clip}}s$.*

The proof is in Appendix A.

Proposition 4.1 states that if the n training samples are embedded in $\mathbb{R}^m$ with S_{clip} as their Gram matrix, and we embed the test sample in $\mathbb{R}^m$ by finding the feature vector whose inner products with the embedded training samples are closest to the given s with respect to the Euclidean distance, then the inner products between the embedded test sample and the embedded training samples are indeed $\tilde{s} = P_{\text{clip}}s$.

On the other hand, there is no matrix multiplication to ensure consistency for spectrum shift. For our experiments using spectrum shift, we adopt the approach of Wu et al. [104], which we mentioned earlier in this section; for this case, their approach is to simply let $\tilde{s} = s$, because spectrum shift only affects self-similarities, and hence there is no change to the off-diagonal elements of the augmented similarity matrix S'.

Table 4.1: Summary of the eight data sets used in the comparative study.

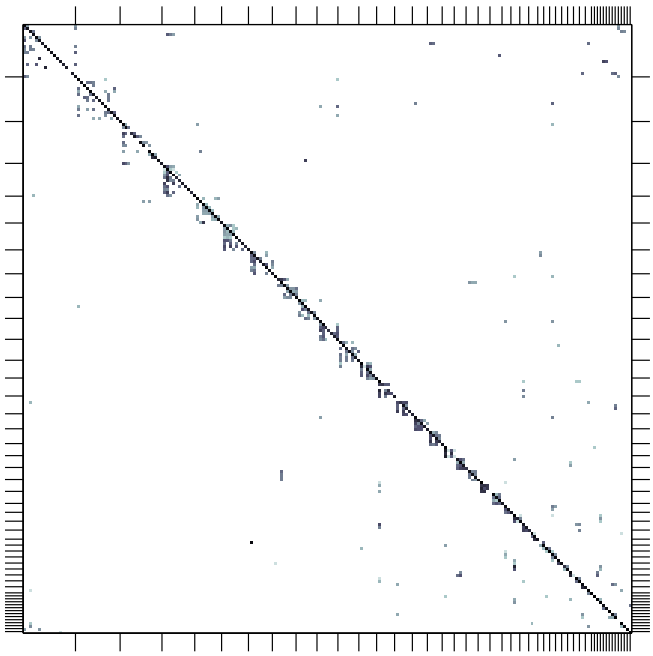
Name	# Samples	# Classes	Symmetric	PSD	Sparse
Amazon-47	204	47	no	no	yes
Aural Sonar	100	2	yes	no	no
Caltech-101	8677	101	yes	yes	no
Face Rec	945	139	yes	no	no
Mirex07	3090	10	yes	no	yes
Patrol	241	8	no	no	yes
Protein	213	4	yes	no	no
Voting	435	2	yes	no	no

4.2 Data Sets

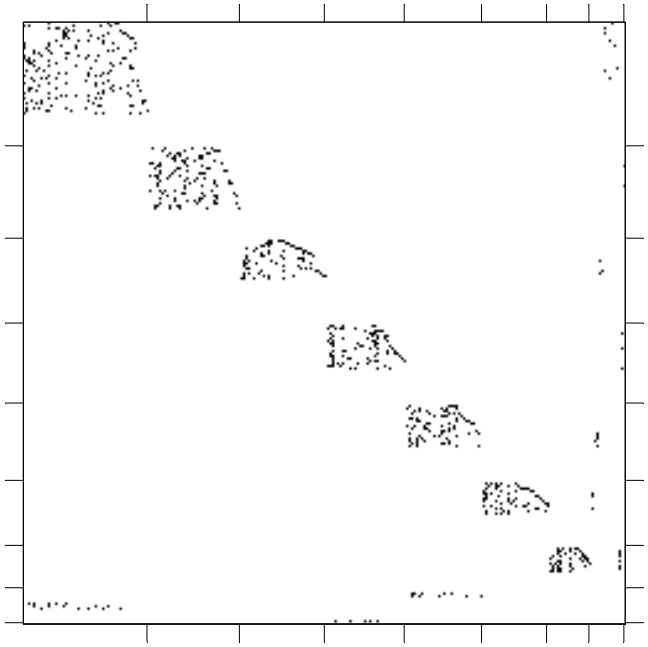
The ten similarity-based classification methods were tested on eight real data sets representing a diverse set of similarities ranging from human judgment of audio signals to sequence alignment scores of proteins. The similarity matrices of these data sets are shown in Figure 4.1–4.5. The rows and columns are ordered by class label; on many of the data sets, particularly those with a fewer number of classes, a block diagonal structure is visible along the class boundaries, indicated by tick marks. Note that a purely block diagonal similarity matrix would indicate a particularly easy classification problem, as objects have nonzero similarities only to objects of the same class. These data sets are described in detail below, and a summary is given in Table 4.1.

The *Amazon-47* data set, shown in Figure 4.1(a), was created for this comparative study, and it consists of 204 books written by 47 authors. Each book listed on `amazon.com` links to the top four books that customers bought after viewing it, along with the percentage of the customers who did so. We take the similarity of book A to book B to be the percentage of the customers who bought book B after viewing book A, and the classification problem is to determine the author of the book.

The *Aural Sonar* data set, shown in Figure 4.2(a), is from a recent paper which inves-

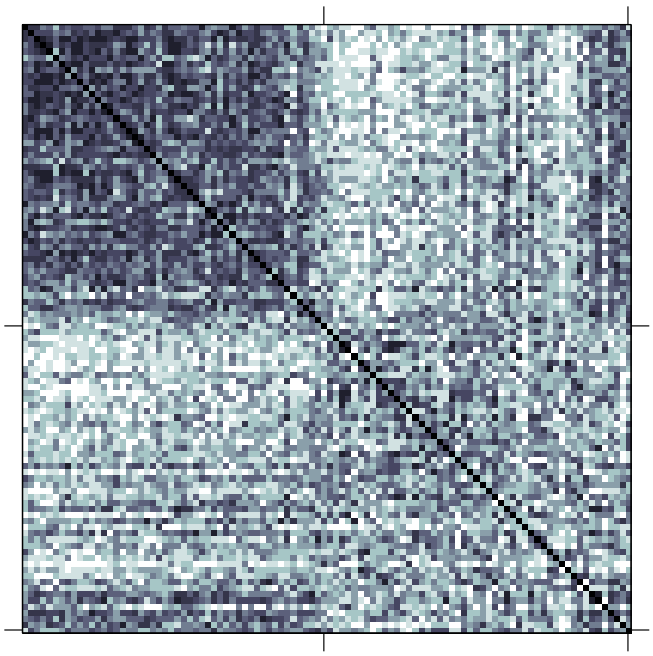


(a) Amazon-47

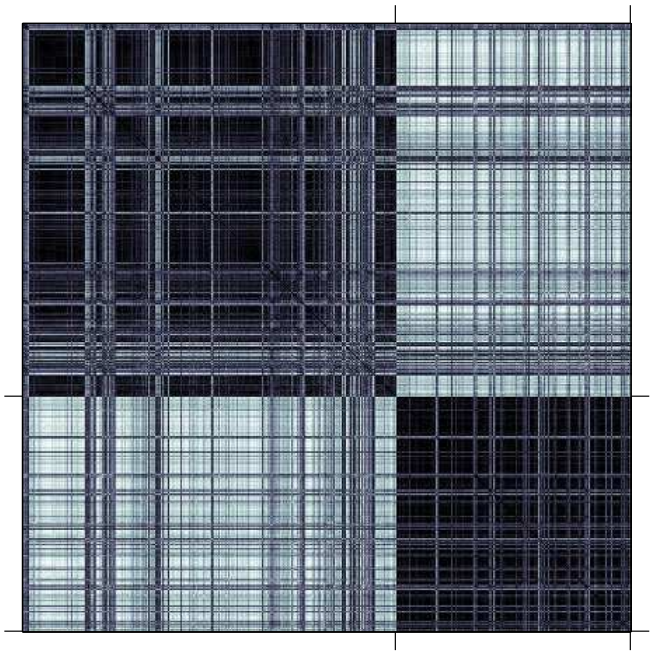


(b) Patrol

Figure 4.1: Similarity matrices of the Amazon-47 and Patrol data sets used in the comparative study. For each matrix, class divisions are indicated by tick marks, and black corresponds to maximum similarity while white corresponds to zero similarity.



(a) Aural Sonar



(b) Voting

Figure 4.2: Similarity matrices of the Aural Sonar and Voting data sets used in the comparative study. For each matrix, class divisions are indicated by tick marks, and black corresponds to maximum similarity while white corresponds to zero similarity.

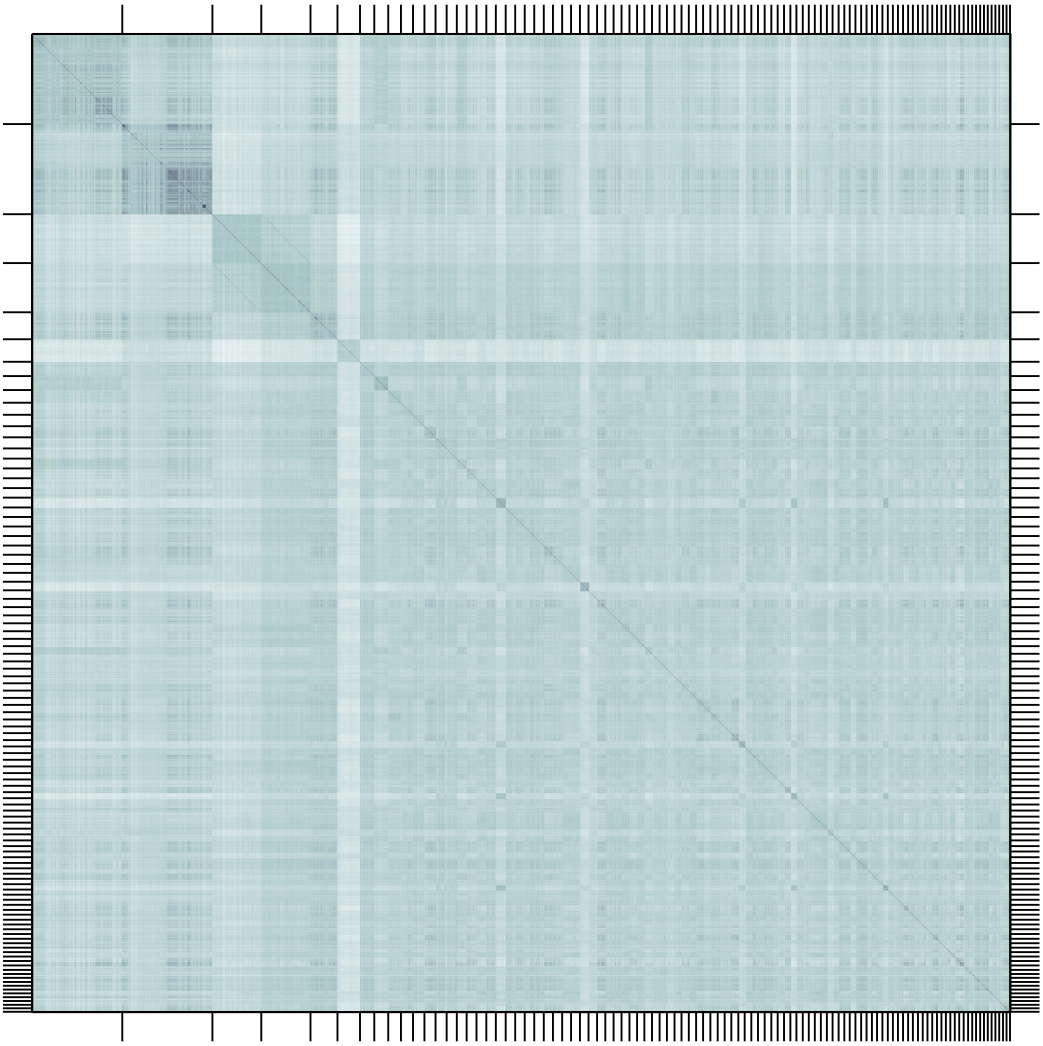
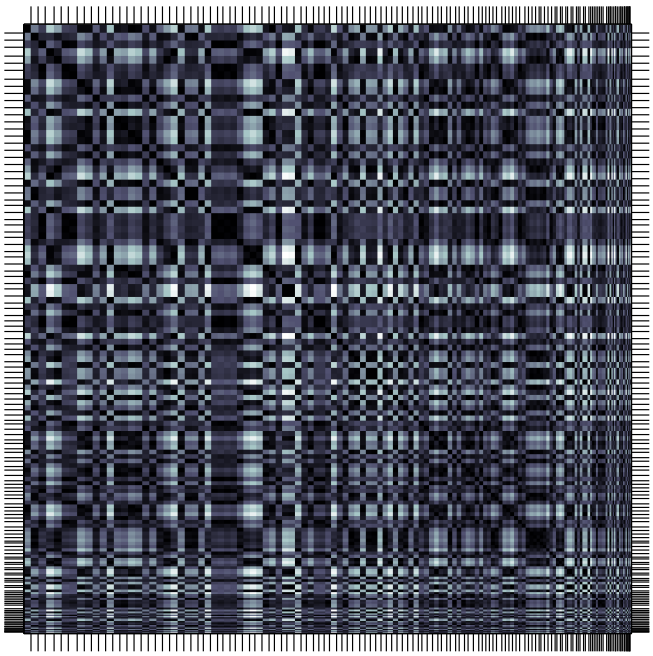
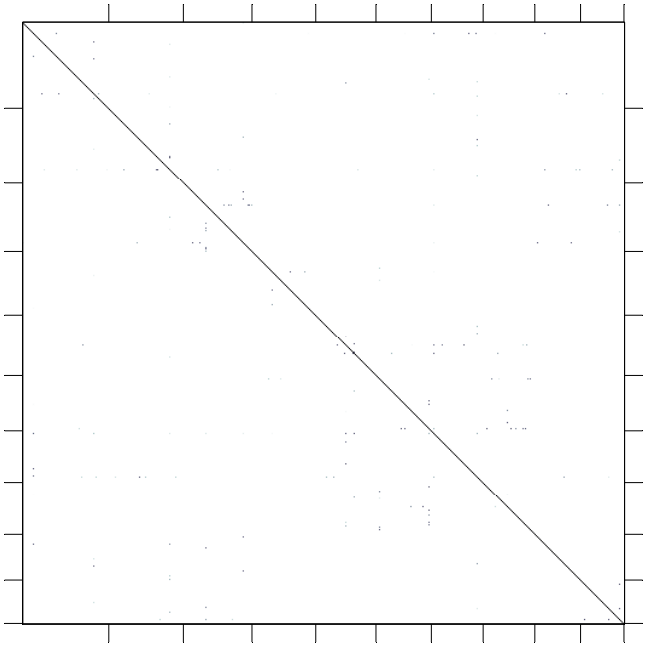


Figure 4.3: Similarity matrix of the Caltech-101 data set used in the comparative study. For this matrix, class divisions are indicated by tick marks, and black corresponds to maximum similarity while white corresponds to zero similarity.

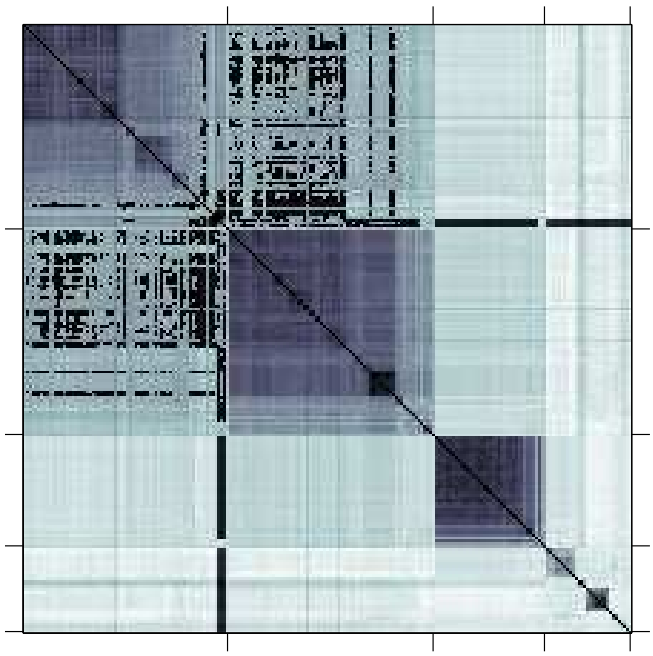


(a) Face Rec

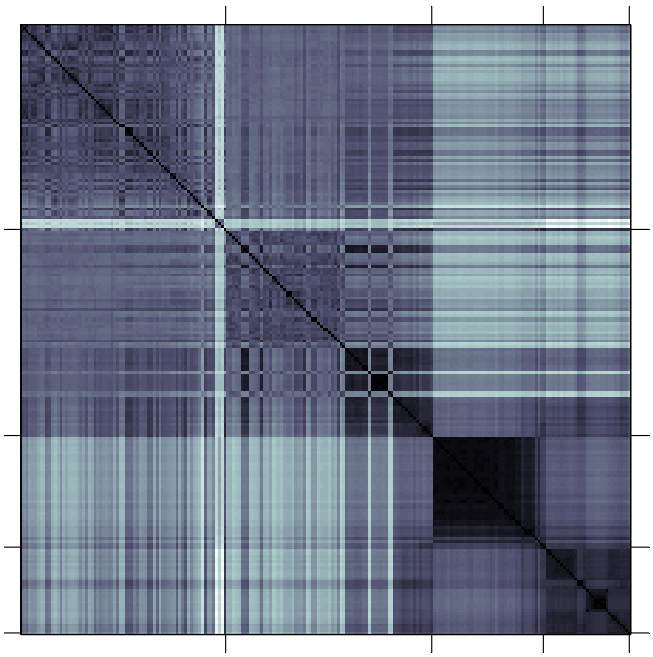


(b) Mirex07

Figure 4.4: Similarity matrices of the Face Rec and Mirex07 data sets used in the comparative study. For each matrix, class divisions are indicated by tick marks, and black corresponds to maximum similarity while white corresponds to zero similarity.



(a) Protein



(b) Protein RBF-sim

Figure 4.5: Similarity matrices of the Protein and Protein RBF-sim data sets used in the comparative study. For each matrix, class divisions are indicated by tick marks, and black corresponds to maximum similarity while white corresponds to zero similarity.

tigated the human ability to distinguish different types of sonar signals by ear [76]. The signals were returns from a broadband active sonar system, with 50 target-of-interest signals and 50 clutter signals. Every pair of signals was assigned a similarity score from 1 to 5 by two randomly chosen human subjects unaware of the true labels, and the two scores were added to produce a 100×100 similarity matrix with integer values from 2 to 10.

The *Caltech-101* data set, shown in Figure 4.3, is an object recognition benchmark data set consisting of 8677 images from 101 object categories [28]. Similarities between images were computed using the pyramid match kernel [40] on SIFT features [63]. Here the similarity is PSD.

The *Face Rec* data set, shown in Figure 4.4(a), consists of 945 sample faces of 139 people from the NIST Fact Recognition Grand Challenge data set.[2] There are 139 classes, one for each person. Similarities for pairs of the original three-dimensional face data were computed as the cosine similarity between integral invariant signatures based on surface curves of the face [29]. The original paper demonstrated comparable results to the state-of-the-art using these similarities with a 1-NN classifier.

The *Mirex07* data set, shown in Figure 4.4(b), was obtained from the human-rated, fine-scale audio similarity data used in the MIREX 2007 Audio Music Similarity and Retrieval evaluation task.[3] Mirex07 consists of 3090 samples, divided roughly evenly among 10 classes that correspond to different music genres. Humans judged how similar two songs are on a 0–10 scale with 0.1 increments. Each song pair was evaluated by three people, and the three similarity values were averaged. Self-similarity was assumed to be 10, the maximum similarity. The classification task is to correctly label each song with its genre.

The *Patrol* data set, shown in Figure 4.1(b), was collected by Driskell and McDonald [25]. Members of seven patrol units were asked to name five members of their unit; in some cases the respondents inaccurately named people who were not in their unit, including people who did not belong to any unit. Of the original 385 respondents and named people, only the ones that were named at least once were kept, reducing the data set to 241

[2] See `http://face.nist.gov/frgc/`

[3] See `http://www.music-ir.org/mirex/wiki/2007:Audio_Music_Similarity_and_Retrieval`

samples. The similarity between any two people a and b is

$$\psi(a,b) = \frac{1}{2}\left(N(a,b) + N(b,a)\right),$$

where $N(a,b)$ is the number of times person a names person b. Thus, here the similarity ψ has range $\{0, 0.5, 1\}$. The classification problem is to estimate to which of the seven patrol units a person belongs, or to correctly place them in an eighth class that corresponds to "not in any of the patrol units."

The *Protein* data set, shown in Figure 4.5(a), has sequence alignment similarities for 213 proteins from 4 classes,[4] where class one through four contains 72, 72, 39, and 30 samples, respectively [46]. As further discussed in the results, we define an additional similarity termed *RBF-sim* for the Protein data set:

$$\psi_{\text{RBF}}(x,x') = \exp\left(-\|s(x) - s(x')\|_2\right),$$

where $s(x)$ is the 213×1 similarity vector with ith component $\psi(x, x_i)$. Its similarity matrix is shown in Figure 4.5(b).

The *Voting* data set, shown in Figure 4.2(b), comes from the UCI Machine Learning Repository [4]. It is a binary classification problem with 435 samples, where each sample is a categorical feature vector with 16 components and three possibilities for each component. We compute the value difference metric (VDM) [93] from the categorical data, which is a dissimilarity that uses the training class labels to weight different components differently so as to achieve maximum probability of class separation. We normalize the dissimilarities such that $d(x,x') \in [0,1]$, and convert them to similarities by letting $\psi(x,x') = 1 - d(x,x')$.

4.3 Experimental Setup

For each data set, we randomly selected 20% of the data for testing and used the remaining 80% for training. The classifier parameters such as C for the SVM, λ for the KRI and KRR weights, and k for local classifiers were chosen by 10-fold cross-validation on the training set, and then the selected parameters were used to train the classifiers on the complete

[4]The original data set has 226 samples with 9 classes. As is standard practice with this data set, we removed those classes which contain less than 7 samples.

Table 4.2: Cross-validation parameter choices for the comparative study.

All local methods	k :	1, 2, 3, . . ., 16, 32, 64, 128
KRI	λ :	10^{-6}, 10^{-5}, . . ., 10, and 10^{6}
KRR	λ :	10^{-3}, 10^{-2}, . . ., 10
PSVM	ϵ :	10^{-4}, 10^{-3}, . . ., 10
PSVM	C :	10^{0}, 10^{1}, . . ., 10^{4}
SVM-KNN	C :	10^{-3}, 10^{-2}, . . ., 10^{5}
SVM (linear, clip, flip, shift)	C :	10^{-3}, 10^{-2}, . . ., 10^{5}
SVM (RBF)	C :	10^{-3}, . . ., 10
SVM (RBF)	γ :	10^{-5}, 10^{-4}, . . ., 10

training data and classify the held out test data. This process was repeated for 20 random partitions of test and training data, and the statistical significance of the classification error was computed by a one-sided Wilcoxon signed-rank test. Multiclass implementations of the SVM classifiers used the "one-vs-one" scheme [48].

Nearest neighbors for local methods were determined using symmetrized similarities,[5] that is, $\frac{1}{2}\left(\psi(x, x_i) + \psi(x_i, x)\right)$. Cross-validation choices are listed in Table 4.2, including those for the parameter γ for the RBF kernel:

$$K_{\text{RBF}}(x, x') = \exp\left(-\gamma \|s - s'\|_2^2\right),$$

where s and s' are the similarity vectors of x and x', respectively. These choices were based on recommendations and usage in previous literature, and on preliminary experiments that we conducted with a larger range of cross-validation parameters on the Voting and Protein data sets.

The main experimental results are shown in Table 4.3, 4.4, and 4.5, respectively. For algorithms that require a PSD similarity matrix S, we made S PSD by spectrum clip, flip and shift as discussed in Section 2.1, and pinv for the KRR weights. The main results,

[5]Only Amazon-47 and Patrol are natively asymmetric.

Table 4.3: Mean and standard deviation (in parentheses) of the test errors (in percentage) across 20 randomized test/training partitions for the Amazon-47, Aural Sonar and Caltech-101 data sets. For each data set, the lowest mean error and those not statistically significantly worse are boldfaced. The six local classifiers are grouped together, and so are the four global classifiers.

	Amazon-47		Aural Sonar		Caltech-101	
k-NN	16.95	(4.85)	17.00	(7.65)	41.55	(0.95)
affinity *k*-NN	**15.00**	(4.77)	**15.00**	(6.12)	39.20	(0.86)
KRI *k*-NN (clip)	17.68	(4.75)	**14.00**	(6.82)	30.13	(0.42)
KRR *k*-NN (pinv)	**16.10**	(4.90)	15.25	(6.22)	**29.90**	(0.44)
Local SDA	16.83	(5.11)	17.75	(7.66)	41.99	(0.52)
SVM-KNN (clip)	17.56	(4.60)	**13.75**	(7.40)	36.82	(0.60)
SVM, sim-as-kernel (clip)	81.34	(4.77)	**13.00**	(5.34)	33.49	(0.78)
SVM, sim-as-features (linear)	76.10	(6.92)	**14.25**	(6.94)	38.18	(0.78)
SVM, sim-as-features (RBF)	75.98	(7.33)	**14.25**	(7.46)	38.16	(0.75)
P-SVM	70.12	(8.82)	**14.25**	(5.97)	34.23	(0.95)

discussed in the next section, are for spectrum clip; the experimental differences between spectrum clip, flip and shift, and pinv are shown in Table 4.6 and discussed in Section 4.5.

4.4 Experimental Results

The mean and standard deviation of the test errors across the 20 randomized test/training partitions are shown in Table 4.3 for the Amazon-47, Aural Sonar and Caltech-101 data sets, in Table 4.4 for the Face Rec, Mirex07 and Patrol data sets, and in Table 4.5 for the Protein, Protein RBF-sim and Voting data sets. The bold results in each column indicate the classifier with the lowest average error; also boldfaced are any classifiers that were not statistically significantly worse than the classifier with the lowest average error.

The similarity matrices of the Aural Sonar and Voting data sets, shown in Figure 4.2, exhibit fairly nice block diagonal structures, indicating that these are somewhat easy classification problems. This is reflected in the relatively low error rates across the board and

Table 4.4: Mean and standard deviation (in parentheses) of the test errors (in percentage) across 20 randomized test/training partitions for the Face Rec, Mirex07 and Patrol data sets. For each data set, the lowest mean error and those not statistically significantly worse are boldfaced. The six local classifiers are grouped together, and so are the four global classifiers.

	Face Rec		Mirex07		Patrol	
k-NN	**4.23**	(1.43)	61.21	(1.97)	**11.88**	(4.42)
affinity *k*-NN	**4.23**	(1.48)	61.15	(1.90)	**11.67**	(4.08)
KRI *k*-NN (clip)	**4.15**	(1.32)	61.20	(2.03)	**11.56**	(4.54)
KRR *k*-NN (pinv)	4.31	(1.86)	61.18	(1.96)	12.81	(4.62)
Local SDA	4.55	(1.67)	60.94	(1.94)	**11.77**	(4.62)
SVM-KNN (clip)	4.23	(1.25)	61.25	(1.95)	**11.98**	(4.36)
SVM, sim-as-kernel (clip)	4.18	(1.25)	57.83	(2.05)	38.75	(4.81)
SVM, sim-as-features (linear)	4.29	(1.36)	**55.54**	(2.52)	42.19	(5.85)
SVM, sim-as-features (RBF)	**3.92**	(1.29)	**55.72**	(2.06)	40.73	(5.95)
P-SVM	**4.05**	(1.44)	63.81	(2.70)	40.42	(5.94)

Table 4.5: Mean and standard deviation (in parentheses) of the test errors (in percentage) across 20 randomized test/training partitions for the Protein, Protein RBM-sim and Voting data sets. For each data set, the lowest mean error and those not statistically significantly worse are boldfaced. The six local classifiers are grouped together, and so are the four global classifiers.

	Protein		Protein RBF		Voting	
k-NN	29.88	(9.96)	**0.93**	(1.71)	5.80	(1.83)
affinity *k*-NN	30.81	(6.61)	**0.93**	(1.71)	5.86	(1.78)
KRI *k*-NN (clip)	30.35	(9.71)	**1.05**	(1.72)	**5.29**	(1.80)
KRR *k*-NN (pinv)	9.53	(5.04)	**1.05**	(1.72)	**5.52**	(1.69)
Local SDA	17.44	(6.52)	**0.93**	(1.71)	6.38	(2.07)
SVM-KNN (clip)	11.86	(5.50)	**1.16**	(1.72)	**5.23**	(2.25)
SVM, sim-as-kernel (clip)	5.35	(4.60)	**1.16**	(1.72)	**4.89**	(2.05)
SVM, sim-as-features (linear)	3.02	(2.76)	2.67	(2.12)	5.40	(2.03)
SVM, sim-as-features (RBF)	**2.67**	(2.97)	2.44	(2.60)	5.52	(1.77)
P-SVM	**1.86**	(1.89)	**1.05**	(1.56)	**5.34**	(1.72)

few statistically significant differences in classification performance. More interesting results can be observed on the more difficult classification problems posed by the other data sets.

The Amazon-47 data set is very sparse with at most four nonzero similarities per row. With such sparse data, one might expect a 1-NN classifier to perform well; indeed, for the uniformly weighted k-NN classifier, the cross-validation chose $k = 1$ on all 20 of the randomized partitions. For all of the local classifiers, the k chosen by cross-validation on this data set was never larger than $k = 3$, and out of the 20 randomized partitions, $k = 1$ was chosen the majority of the time for all the local classifiers. In contrast, the global SVM classifiers perform poorly on this sparse data set. The patrol data set is the next sparsest data set, and the results show a similar pattern, that is, local classifiers on average perform significantly better than global classifiers. However, the Mirex07 data set is also relatively sparse, and yet the global classifiers do well, in particular the SVMs that use similarities as features. The Amazon-47 and Patrol data sets do differ from the Mirex07 data set in that the self-similarities are not always maximal, and also samples in the Mirex07 data set tend to have strong interclass similarities more often than those in the other two sparse data sets. Whether and how either of these two differences causes or correlates the relative differences in classification performance is worth further investigation.

On the Protein data set, classifiers exhibit large differences in their performance, with the statistically significantly lowest error rates achieved by two of the three SVMs that use similarity as features. The reason why using similarities as features performs so well while others do so poorly is because the first and second classes exhibit a strong interclass similarity, and rows belonging to the same class exhibit very similar patterns, thus treating the rows of the similarity matrix as feature vectors provides good discrimination of classes. To investigate this effect, we transformed the entire data set using a Gaussian radial basis function (RBF) to create a similarity based on the Euclidean distance between rows of the original similarity matrix, yielding the 213×213 Protein RBF similarity matrix. One can see from Table 4.5 that this transformation increases the performance of classifiers across the board, indicating that this is indeed a better measure of similarity for this particular data set. Furthermore, after this transformation, we see a complete turnaround in perfor-

mance: for Protein RBF-sim, the SVMs that use similarities as features are among the worst performers (with the P-SVM still performing decently), and the vanilla k-NN does better than the best classifier given the original Protein similarities.

The Caltech-101 data set is the largest data set, and with 8677 samples in 101 classes, an analysis of the structure of this similarity matrix is difficult. Here we see the most dramatic improvement in classification performance (25% lower error) by using the KRR and KRI weights rather than k-NN with the uniform or affinity weights, suggesting that there are highly correlated samples that bias the classification, which might be related to the fact that the original set of images was collected using Google Images[6] [28]. In contrast, for the Amazon-47, Aural Sonar, Face Rec, Mirex07, and Patrol data sets, there is only a small win by using the KRI or KRR weights, or a statistically insignificant small decline in performance (we hypothesize that this occurred because of overfitting due to the additional parameter λ). On the Protein data set, the error rate of the KRR weights is one third of the error of any other k-NN method; this is a consequence of using the pseudoinverse rather than other types of spectrum modification, as can be easily seen from the results in Table 4.6. The other significant difference between the weighting methods is a roughly 10% improvement in average error on the Voting data set by using the KRI or KRR weights. In conclusion, the use of diverse weights may not matter on some data sets, but can be very helpful on certain data sets.

SVM-KNN was proposed by Zhang et al. in part as a way to reduce the computations required to train a global SVM using similarities as a kernel [108], and their results showed that it performed similarly to a global SVM. That is somewhat true here, but some differences emerge. For the Amazon-47 and Patrol data sets, the local methods all do well including SVM-KNN, but the global methods do poorly, including the global SVM using similarities as a kernel. On the other hand, the global SVM using similarities as a kernel is statistically significantly better than SVM-KNN on Caltech-101, even though the best performance on that data set is achieved by a local method (k-NN with the KRR weights). From this sampling of data sets, we conclude that applying the SVM locally or globally can

[6] See `http://images.google.com/`

in fact make a difference, but whether it is a positive or negative difference depends on the application.

Among the four global SVMs (including the P-SVM), it is hard to draw conclusions about the performance of the one that uses similarities as a kernel versus the other three that use similarities as features. In terms of statistical significance, the SVM using similarities as a kernel outperforms the others on the Patrol and Caltech-101 data sets whereas it is the worst on the Amazon-47 and Protein data sets, and there is no clear division on the remaining data sets. Lastly, the results do not show statistically significant differences between using the linear or RBF version of the SVM with similarities as features except for small difference on the Face Rec and Patrol data sets.

4.5 Clip, Flip, or Shift?

Three different approaches to modifying similarities to form a kernel were discussed in Section 2.1. We experimentally compared spectrum clip, flip and shift for the KRI weights, SVM-KNN and SVM, and clip, flip, shift and pinv for the KRR weights on the nine data sets. Table 4.6 shows the results on the four data sets for which at least one method showed statistically significantly different results depending on the choice of spectrum modification method.

For the KRR weights, one can see that the pinv solution is never statistically significantly worse than spectrum clip, flip or shift, which are worse than pinv at least once. For the KRI weights, the differences are negligible, but based on the average error, I recommend clip.

Spectrum flip takes the absolute value of the eigenvalues, which is similar to the effect of spectrum square, that is, using SS^T as a kernel (as discussed in Section 2.1.6), which for an SVM is equivalent to treating similarities as features. Therefore it is not surprising that for the Protein data set, which, according to the results in Table 4.5, favors treating similarities as features, flip makes a large positive difference for both SVM-KNN and SVM.

Lastly, notice from Table 4.6 the different effects of the spectrum modifications on the local methods versus the global SVM. This is because for the local methods, the modifi-

Table 4.6: Clip, flip, shift, and pinv comparison. The following table shows the test errors (in percentage) averaged over 20 randomized test/training partitions for the four data sets that exhibit statistically significant differences between these spectrum modifications. If there are statistically significant differences for a given algorithm and a given data set, then the highest error and those not statistically significantly better are boldfaced.

		Amazon-47	Mirex07	Patrol	Protein
KRI	clip	17.68	61.20	11.56	30.35
	flip	17.56	61.17	11.67	31.28
	shift	17.68	61.25	**13.23**	30.35
KRR	clip	16.22	**61.22**	11.67	**30.35**
	flip	16.22	61.12	**12.08**	**30.47**
	shift	16.34	**61.25**	11.88	**30.35**
	pinv	16.10	**61.18**	**12.81**	9.53
SVM-KNN	clip	17.56	61.25	11.98	11.86
	flip	17.56	61.25	11.88	1.74
	shift	17.56	61.25	11.88	**30.23**
SVM	clip	81.34	57.83	38.75	5.35
	flip	**84.27**	56.34	**47.29**	1.51
	shift	77.68	**85.29**	40.83	**23.49**

cation is done only locally, which is on a much smaller scale than performing spectrum modification on the similarity matrix of the entire training set.

4.6 Probability Estimates from Weighted k-NN

We mentioned in Chapter 3 that for weighted k-NN, when the weights are nonnegative and sum to one, they can be used to form posterior probability estimates as given by (3.2). In this section, we evaluate the quality of the posterior probability estimates using the uniform weights, the affinity weights and the KRI weights proposed in Section 3.2.

4.6.1 Evaluation Measure

First, we discuss the evaluation measure. For a fixed $x \in \Omega$, let $P(y \mid x)$ and $\hat{P}(y \mid x)$ be the true and estimated posterior probabilities, respectively. To measure how close the posterior probability estimate is to the true one, we can use the Kullback-Leibler (KL) divergence between $P(y \mid x)$ and $\hat{P}(y \mid x)$, that is,

$$D_{\mathrm{KL}}(P(y \mid x) \,\|\, \hat{P}(y \mid x)) = \sum_{y \in \mathcal{G}} P(y \mid x) \log \frac{P(y \mid x)}{\hat{P}(y \mid x)} = E_{Y|x} \left(\log \frac{P(Y \mid x)}{\hat{P}(Y \mid x)} \right).$$

To measure the quality of the posterior probability estimate for all $x \in \Omega$, we need to average the above KL divergence over the entire sample space Ω, that is,

$$\begin{aligned} E_X \left(D_{\mathrm{KL}}(P(y \mid X) \,\|\, \hat{P}(y \mid X)) \right) &= E_X \left(E_{Y|X} \left(\log \frac{P(Y \mid X)}{\hat{P}(Y \mid X)} \right) \right) \\ &= E_{X,Y} \left(\log \frac{P(Y \mid X)}{\hat{P}(Y \mid X)} \right) \\ &= E_{X,Y} \left(\log P(Y \mid X) \right) + E_{X,Y} \left(-\log \hat{P}(Y \mid X) \right) \\ &= -H(Y \mid X) + E_{X,Y} \left(-\log \hat{P}(Y \mid X) \right), \end{aligned}$$

where $H(Y \mid X) = E_{X,Y} \left(-\log P(Y \mid X) \right)$ is the conditional entropy, and we further denote the second term by

$$\zeta \triangleq E_{X,Y} \left(-\log \hat{P}(Y \mid X) \right).$$

Since $H(Y \mid X)$ does not depend on $\hat{P}(y \mid x)$, for any posterior probability estimate $\hat{P}(y \mid x)$, the lower ζ is, the lower the expected KL divergence is, and the closer it is to the true

posterior probability. Hence we can use ζ to measure the quality of $\hat{P}(y \mid x)$. Unfortunately, we do not know the true joint probability distribution of (X, Y) for real data sets, but by applying the law of large numbers, we can obtain the following empirical estimate of ζ:

$$\hat{\zeta} = -\frac{1}{n} \sum_{i=1}^{n} \log \hat{P}(y_i \mid x_i),$$

which is sometimes called the *empirical cross entropy*. In our experiments, we use $\hat{\zeta}$ as the evaluation measure: the lower $\hat{\zeta}$ is, the better we consider $\hat{P}(y \mid x)$ is. Following the tradition of natural language processing [66], we report the results by perplexity, which is defined as

$$\text{perplexity} \triangleq 2^{\hat{\zeta}} = 2^{-\frac{1}{n} \sum_{i=1}^{n} \log_2 \hat{P}(y_i \mid x_i)} = \left(\prod_{i=1}^{n} \hat{P}(y_i \mid x_i) \right)^{-\frac{1}{n}}.$$

The posterior probability estimate with lower perplexity is considered better.

Another measure commonly used in natural language processing for confidence estimation is the *normalized cross entropy* (NCE) [31], which is defined as

$$\text{NCE} \triangleq \frac{\hat{\zeta}_0 - \hat{\zeta}}{\hat{\zeta}_0},$$

where $\hat{\zeta}_0$ is the empirical cross entropy using a baseline probability estimate. Usually, the class prior probability estimated from the training data is used as the baseline model. The normalized cross entropy is bounded above by one, and the larger it is, the better the posterior probability estimate is.

4.6.2 Results

We compared the posterior probability estimates formed by the uniform, affinity and KRI weights on the Aural Sonar, Caltech-101, Face Rec, Patrol and Voting data sets by performing leave-one-out cross-validation, that is, for each sample, we found its k-nearest neighbors from the rest of the data set. We computed $\hat{\zeta}$ for each k that is less than n from the following set:

$$k \in \{1, 2, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96, 128, 192, 256\},$$

and for the KRI weights, we chose the regularization parameter λ from

$$\lambda \in \{10^{-6}, 10^{-5}, \ldots, 1, 10, 100\}.$$

When computing $\hat{\zeta}$, it is possible that for some $i \in \{1, \ldots, n\}$, we have $\hat{P}(y_i \mid x_i) = 0$. This happens to all the three weighting methods when none of the k-nearest neighbors of x_i are from its class y_i. This can also happen when the affinity or KRI weights assign zero weight to all the neighbors of x_i from its class y_i, which is theoretically possible but not very likely in practice. In order to avoid the singular case that $\hat{P}(y_i \mid x_i) = 0$, we used a smoothed version of $\hat{P}(y \mid x)$:

$$\hat{P}_{\text{smooth}}(y \mid x) = \frac{\hat{P}(y \mid x) + \epsilon}{\sum_{g \in \mathcal{G}} \left(\hat{P}(g \mid x) + \epsilon\right)},$$

where $\epsilon > 0$ is a small positive number.[7] The above smoothing is called *additive smoothing* or sometimes *Lidstone smoothing*.

The lowest perplexities achieved by the three weighting methods on the five data sets that we used for this experiment are shown in Table 4.7. It is easy to see that among the three weighting methods, the proposed KRI weights consistently achieved the lowest perplexity, which indicates that with a proper regularization parameter λ, the KRI weights can achieve better posterior probability estimates than both the uniform and affinity weights. Notice that the relatively big reduction in perplexity on the Caltech-101 data set achieved by the KRI weights is consistent with the classification performance reported in Table 4.3.

For completeness, we also show in Table 4.8 the highest normalized cross entropies achieved by the three weighting methods on the five data sets, where for each sample, the class prior probability estimated from the rest of the data set is used as the baseline model. The normalized cross entropies lead to the same conclusion as do the perplexities.

We plot the value of the empirical cross entropy $\hat{\zeta}$ versus the neighborhood size k in Figure 4.6–4.8. For the KRI weights, the λ's of the plots are those that helped achieve the lowest perplexities shown in Table 4.7. In general, when k increases, it is more likely that a sample x will have neighbors from other classes; when weights are assigned to these

[7] We chose $\epsilon = 10^{-4}$ for the Aural Sonar, Patrol and Voting data sets, and $\epsilon = 10^{-6}$ for the Caltech-101 and Face Rec data sets. These numbers were determined by the number of classes in each data set and several preliminary experiments.

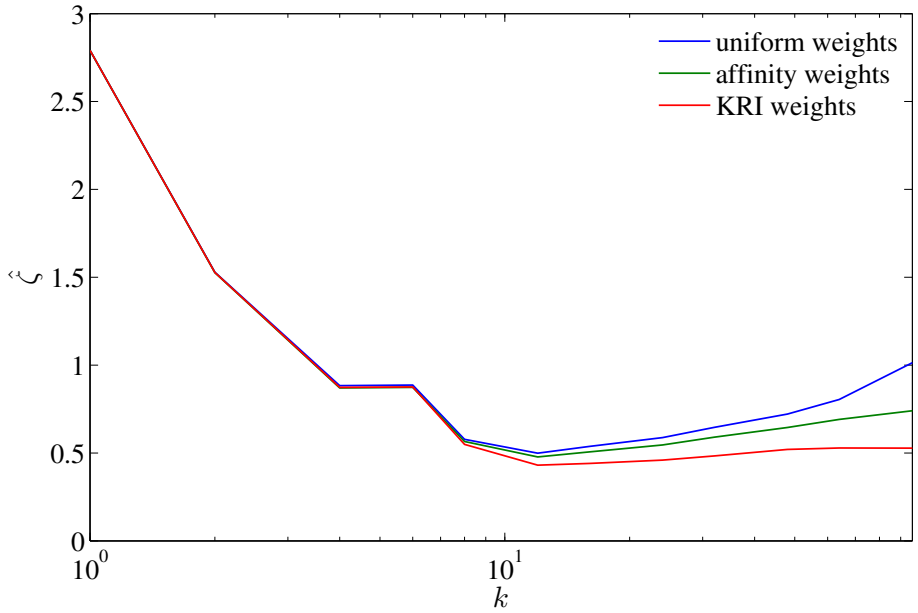
Table 4.7: Lowest perplexities achieved by the uniform, affinity and KRI weights during the leave-one-out cross-validation.

	Aural Sonar	Caltech-101	Face Rec	Patrol	Voting
uniform weights	1.41	10.24	1.32	1.92	1.14
affinity weights	1.39	9.97	1.32	1.82	1.14
KRI weights	1.35	6.45	1.18	1.70	1.12

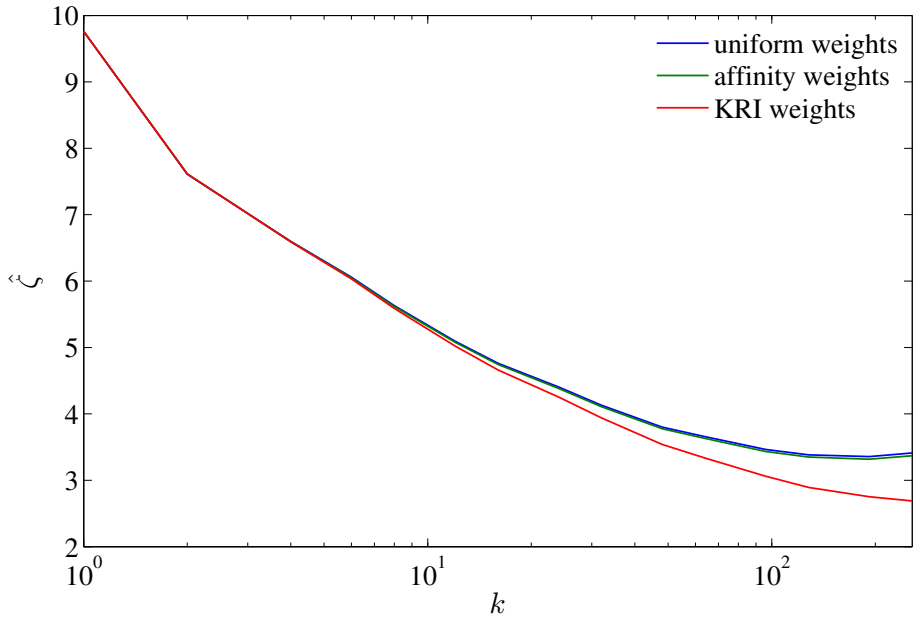
Table 4.8: Highest normalized cross entropies achieved by the uniform, affinity and KRI weights during the leave-one-out cross-validation.

	Aural Sonar	Caltech-101	Face Rec	Patrol	Voting
uniform weights	0.5083	0.4459	0.9435	0.6819	0.7984
affinity weights	0.5295	0.4522	0.9436	0.7072	0.7994
KRI weights	0.5757	0.5559	0.9658	0.7406	0.8243

neighbors, the estimated probability of x's own class goes down, which explains why for the uniform and affinity weights, $\hat{\zeta}$ goes up when k becomes large. An exception can be observed in Figure 4.7(b) for the affinity weights on the Patrol data set. This is because the similarity matrix of the Patrol data set is sparse (see Figure 4.1); for a large k, many "neighbors" actually have zero similarity to x, and therefore will be assigned zero weight according to the affinity weights. For the KRI weights, we did not observe a dramatic increase of $\hat{\zeta}$ when k goes large. This is a result of the diversity goal: when those neighbors from other classes are more similar to each other than to the sample x, they will not receive much weight from the KRI weighting method. In other words, by exploring the similarity structure among neighbors, the KRI weights gain more information and hence produce better posterior probability estimates.

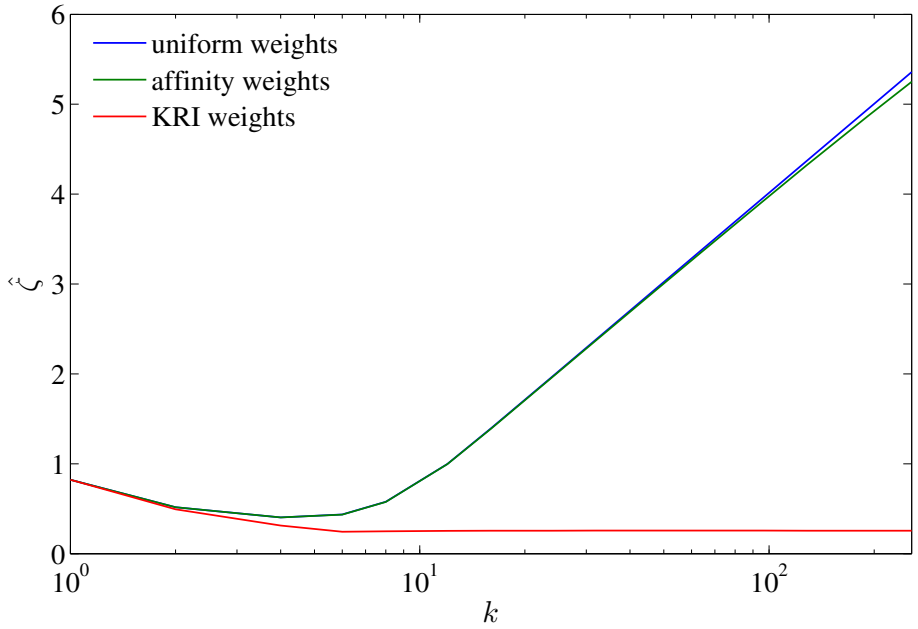


(a) Aural Sonar

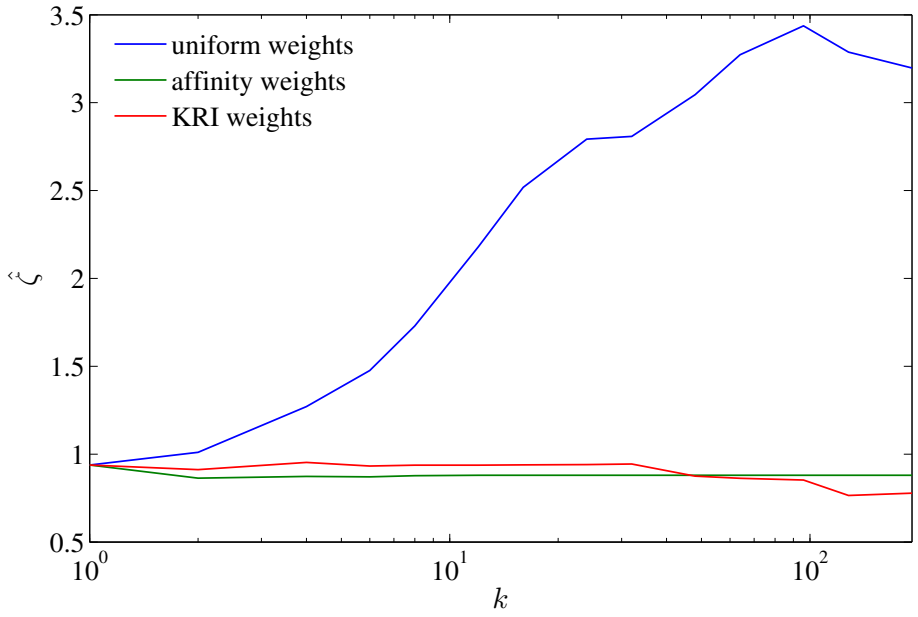


(b) Caltech-101

Figure 4.6: How $\hat{\zeta}$ changes with the neighborhood size k on the Aural Sonar and Caltech-101 data sets are shown in (a) and (b), respectively. Here the KRI weights use $\lambda = 1$ and $\lambda = 10^{-3}$ for the Aural Sonar and Caltech-101 data sets, respectively.



(a) Face Rec



(b) Patrol

Figure 4.7: How $\hat{\zeta}$ changes with the neighborhood size k on the Face Rec and Patrol data sets are shown in (a) and (b), respectively. Here the KRI weights use $\lambda = 10^{-4}$ and $\lambda = 10^{-1}$ for the Face Rec and Patrol data sets, respectively.

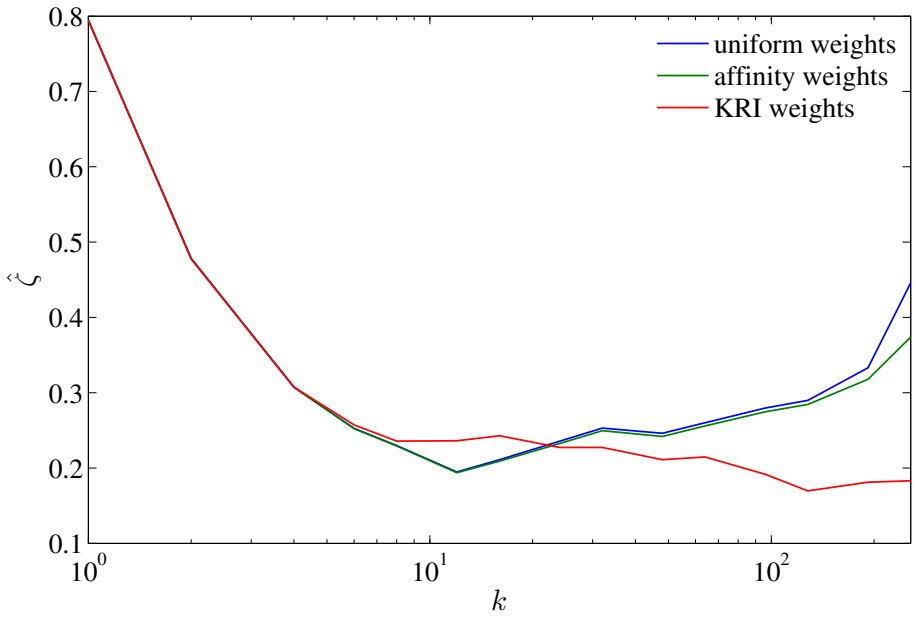


Figure 4.8: The above plot shows how $\hat{\zeta}$ changes with the neighborhood size k on the Voting data set. Here the KRI weights use $\lambda = 10^{-1}$.

Chapter 5

LEARNING KERNELS FROM INDEFINITE SIMILARITIES

Joint undertakings stand a better chance
when they benefit both sides.
Euripides

The results in Table 4.6 show that different approaches to modifying indefinite similarities into kernels can yield different results. This motivates me to investigate directly learning a kernel given indefinite similarities that produces a classifier with good generalization. Formally, given a similarity function $\psi(x, x')$, we would like to seek a surrogate kernel function $K(x, x')$ that induces an RKHS in which a classifier with good generalization can be learned. However, assuming one only has access to the values of the similarity function for all pairs of the training samples, I pose the problem as follows:

> Given an indefinite similarity matrix S, can we find a surrogate PSD matrix K corresponding to an RKHS in which a classifier with good generalization can be learned?

In Section 5.1, we first consider the surrogate kernel matrix K to be a free parameter in the SVM primal problem. Then in Section 5.2, we restrict the surrogate kernel matrix K to be a spectrum modification of S in order to reduce overfitting and yield a more tractable optimization problem. Some experimental results are shown in Section 5.3, and in Section 5.4, we discuss an extension that combines indefinite similarities with multiple kernels. A more concise version of some of the research in this chapter appears in my publications [23, 22].

5.1 Learning the Kernel Matrix

The traditional way of using indefinite similarities for the SVM decouples the steps of modifying indefinite similarities into a kernel and training an SVM using that kernel. Basically,

the modification does not take the class labels of the data into account. Since the training of the SVM depends on the kernel, which is a result of the modification, it is very natural to ask: Can we directly learn a kernel from the data in a more principled way? And can we do it together with learning an SVM?

To answer these two questions, we consider minimizing the empirical risk with regularization jointly over the kernel matrix K and the original SVM parameters, that is,

$$\underset{f,K}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} L_{\text{hinge}}(y_i, f(x_i)) + \eta\|f\|_K^2 + \gamma\|K - S\|_F, \tag{5.1}$$

where f has the form

$$f(x) = \sum_{i=1}^{n} c_i K(x, x_i) + b,$$

and $\eta > 0$ and $\gamma > 0$ are both regularization parameters. Comparing (5.1) with (2.1), we added an extra regularization term $\|K - S\|_F$, which focuses the search for K in the vicinity of S in terms of the Frobenius norm. This regularization term imposes the belief that a good kernel matrix K is close to the given indefinite similarity matrix S; however, if there is different prior knowledge about how K and S are related, one can easily replace $\|K - S\|_F$ with a term that more appropriately reflects the known relationship between K and S.

For the development of the investigated methods, we favor the primal form of the SVM as given in (2.3) due to its clear mathematical interpretation in terms of empirical risk minimization with regularization. To that end, we reformulate (5.1) as an extension of the primal form of the SVM given in (2.3):

$$\begin{aligned} \underset{c,b,\xi,K}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta c^T K c + \gamma\|K - S\|_F \\ \text{subject to} \quad & \text{diag}(y)(Kc + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & K \succeq 0, \end{aligned} \tag{5.2}$$

with additional variable $K \in \mathbb{R}^{n\times n}$ and additional regularization parameter $\gamma > 0$. Recall from Section 2.1.3 that spectrum clip yields the nearest PSD matrix to S in term of the Frobenius norm; hence, when γ is set very large, solving (5.2) is almost the same as training an SVM with S_{clip}.

It is not trivial to solve (5.2) as formulated; the optimization problem given by (5.2) is not convex because both c and K are variables and we have a quadratic form $c^T Kc$ in the objective and a product term Kc in one of the constraints. We show that by using the following lemma, (5.2) can in fact be expressed as a convex conic program, whose special structure enables us to design efficient algorithms [70], and for small to medium size data sets, we can solve the problem using a general-purpose convex conic optimizer such as SeDuMi [94] or SDPT3 [97].

Lemma 5.1. *Let $K \in \mathbb{R}^{n\times n}$, $z \in \mathbb{R}^n$ and $u \in R$. Then*

$$\begin{bmatrix} K & z \\ z^T & u \end{bmatrix} \succeq 0$$

if and only if $K \succeq 0$, z is in the range (column space) of K, and $u - z^T K^\dagger z \geq 0$, where $K^\dagger$ is the Moore-Penrose pseudoinverse of K.

Lemma 5.1 follows directly from [47, p. 44, Theorem 1.20], which states a basic property of the generalized Schur complement. Note that Lemma 5.1 can be concisely expressed as

$$\begin{bmatrix} K & z \\ z^T & u \end{bmatrix} \succeq 0 \;\Leftrightarrow\; K \succeq 0,\; z = Kc,\; u - z^T K^\dagger z \geq 0, \tag{5.3}$$

where $z = Kc$ for some $c \in \mathbb{R}^n$ is equivalent to saying that z is in the range of K. Since the pseudoinverse of K satisfies the following property

$$KK^\dagger K = K,$$

for $z = Kc$, we have

$$z^T K^\dagger z = c^T KK^\dagger Kc = c^T Kc,$$

and hence (5.3) is equivalent to

$$\begin{bmatrix} K & z \\ z^T & u \end{bmatrix} \succeq 0 \;\Leftrightarrow\; K \succeq 0,\; z = Kc,\; c^T Kc \leq u. \tag{5.4}$$

The equivalence condition given by (5.4) is the key to the reformulation of (5.2).

In order to formulate (5.2) as a convex conic program, we first introduce slack variables $u, v \in \mathbb{R}$, and rewrite (5.2) as

$$\begin{aligned} \underset{c,b,\xi,K,u,v}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta u + \gamma v \\ \text{subject to} \quad & \operatorname{diag}(y)(Kc + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & K \succeq 0, \\ & c^T K c \leq u, \\ & \|K - S\|_F \leq v. \end{aligned} \tag{5.5}$$

Then we let $z = Kc$, and based on the equivalence condition given by (5.4), we can easily rewrite (5.5) as the following equivalent problem:

$$\begin{aligned} \underset{z,b,\xi,K,u,v}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta u + \gamma v \\ \text{subject to} \quad & \operatorname{diag}(y)(z + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & \begin{bmatrix} K & z \\ z^T & u \end{bmatrix} \succeq 0, \\ & \|K - S\|_F \leq v, \end{aligned} \tag{5.6}$$

with variables $z \in \mathbb{R}^n$, $b \in \mathbb{R}$, $\xi \in \mathbb{R}^n$, $K \in \mathbb{R}^{n\times n}$, $u \in \mathbb{R}$ and $v \in \mathbb{R}$. The problem in (5.6) is a convex conic program since it has a linear objective, a set of affine constraints, a linear matrix inequality (LMI) constraint and a second-order cone (SOC) constraint.

After solving (5.6), we obtain the optimal solution, denoted by $z^\star$, $b^\star$ and $K^\star$. Here $K^\star$ is the kernel matrix learned from the data, and $b^\star$ is the bias term for the SVM, but we still need to recover the SVM parameter $c \in \mathbb{R}^n$. Again, by Lemma 5.1, we know that there must exist a $c \in \mathbb{R}^n$ that satisfies

$$z^\star = K^\star c. \tag{5.7}$$

Such c is unique when $K^\star$ is full-rank, but when $K^\star$ is not full-rank, such c is not unique. In fact, when $K^\star$ is not full-rank, the null space of $K^\star$, denoted by $\operatorname{null}(K^\star)$, contains nonzero

vectors. Suppose c is a solution to (5.7), then for any $a \in \text{null}(K^\star)\backslash\{0\}$, $c' = c + a$ is still a solution to (5.7), and both c and c' are optimal solutions to (5.2). In order to avoid the ambiguity when recovering c, we choose the optimal $c^\star$ to be

$$c^\star = (K^\star)^\dagger z^\star,$$

which, according to the properties of the Moore-Penrose pseudoinverse, solves the following least-norm problem:

$$\begin{aligned} \underset{c}{\text{minimize}} \quad & \|c\|_2 \\ \text{subject to} \quad & K^\star c = z^\star, \end{aligned}$$

or in other words, $c^\star$ is the solution to (5.7) with minimum Euclidean norm [68].

5.1.1 Related Work: Indefinite SVM

The idea of jointly learning a kernel matrix and training an SVM first appeared in [64]. This paper took the perspective that indefinite similarities are noisy observations of an unknown PSD kernel; based on this perspective, the authors of [64] considered all the PSD matrices within distance β to S, that is,

$$\mathcal{K}_{S,\beta} = \left\{ K \succeq 0 \;\middle|\; \|K - S\|_F \leq \beta \right\},$$

where $\beta > \min_{K\succeq 0} \|K - S\|_F$, and proposed to maximize the minimum of the SVM dual objective among these matrices:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \min_{K \in \mathcal{K}_{S,\beta}} \left(\mathbf{1}^T\alpha - \frac{1}{2}\alpha^T \operatorname{diag}(y) K \operatorname{diag}(y) \alpha \right) \\ \text{subject to} \quad & 0 \leq \alpha \leq C\mathbf{1}, \\ & y^T\alpha = 0. \end{aligned}$$

In practice, they replaced the hard constraint $\|K - S\|_F \leq \beta$ with a penalty term and proposed the following problem for indefinite similarity matrix S:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \min_{K \succeq 0} \left(\mathbf{1}^T\alpha - \frac{1}{2}\alpha^T \operatorname{diag}(y) K \operatorname{diag}(y) \alpha + \rho\|K - S\|_F^2 \right) \\ \text{subject to} \quad & 0 \leq \alpha \leq C\mathbf{1}, \\ & y^T\alpha = 0, \end{aligned} \tag{5.8}$$

where $\rho > 0$ is the parameter to control the trade-off. They referred to (5.8) as the *Indefinite SVM*. They pointed out that the inner problem of (5.8) has a closed-form solution, that is, for a fixed α,

$$\begin{aligned} &\arg\min_{K \succeq 0} \left(-\frac{1}{2} \alpha^T \operatorname{diag}(y) K \operatorname{diag}(y) \alpha + \rho \|K - S\|_F^2 \right) \\ &= \left(S + \frac{1}{4\rho} \left(\operatorname{diag}(y)\alpha \right) \left(\operatorname{diag}(y)\alpha \right)^T \right)_{\text{clip}}, \end{aligned} \tag{5.9}$$

where $(\cdot)_{\text{clip}}$ denotes the spectrum clip operator. They also pointed out that outer problem is convex since its objective is a pointwise minimum of a set of concave quadratic functions of α and thus also concave in α [13].

The problem in (5.8) looks very different from what we formulated in (5.2); these two problems were formed from completely different perspectives: (5.2) extends the primal form of the SVM by including kernel matrix K as an additional variable, while (5.8) extends the dual form of the SVM by applying the maximin rule to the original objective function. Even philosophically, these two problems seem very different: (5.2) takes an optimistic view of finding a kernel matrix K that results in a better classifier, while (5.8) was interpreted by the authors of [64] as "a worst-case robust classification problem with bounded uncertainty on the kernel matrix K" with a clearly pessimistic overtone. However, with the help of the following theorem, I will show below that these two problems are in fact equivalent except for a slight difference in the regularizer of K.

Theorem 5.2 (Sion's Minimax Theorem)**.** *Let M and N be convex spaces one of which is compact, and $f(\mu, \nu)$ a function on $M \times N$, which is quasiconcave in M, quasiconvex in N, upper semi-continuous in μ for each $\nu \in N$, and lower semi-continuous in ν for each $\mu \in M$, then*

$$\sup_{\mu \in M} \inf_{\nu \in N} f(\mu, \nu) = \inf_{\nu \in N} \sup_{\mu \in M} f(\mu, \nu).$$

This general minimax theorem was proved in [88].

First, we denote the feasible set of the SVM dual problem shown in (2.4) by

$$\mathcal{A} = \left\{ \alpha \in \mathbb{R}^n \,\middle|\, y^T \alpha = 0,\ 0 \le \alpha \le C\mathbf{1} \right\},$$

and rewrite (5.8) as

$$\max_{\alpha \in \mathcal{A}} \min_{K \succeq 0} \ \mathbf{1}^T \alpha - \frac{1}{2} \alpha^T \operatorname{diag}(y) K \operatorname{diag}(y) \alpha + \rho \|K - S\|_F^2. \tag{5.10}$$

Because $\mathcal{A}$ and the PSD cone are both convex, $\mathcal{A}$ is compact, and the objective function of (5.10) is continuous in α and K and is concave in α and convex in K, we apply Theorem 5.2 and switch the max and the min. Therefore, (5.10) is equivalent to

$$\min_{K \succeq 0} \max_{\alpha \in \mathcal{A}} \mathbf{1}^T \alpha - \frac{1}{2} \alpha^T \operatorname{diag}(y) K \operatorname{diag}(y) \alpha + \rho \|K - S\|_F^2. \tag{5.11}$$

Since (2.4) is the dual of (2.3) with zero duality gap, we can replace the (2.4) part in the inner problem of (5.11) with (2.3), then merge the inner minimization over c, b and ξ with the outer minimization over K, and obtain the following equivalent problem of the indefinite SVM:

$$\begin{aligned} \underset{c, b, \xi, K}{\text{minimize}} \quad & \frac{1}{n} \mathbf{1}^T \xi + \eta c^T K c + \rho \|K - S\|_F^2 \\ \text{subject to} \quad & \operatorname{diag}(y)(Kc + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & K \succeq 0. \end{aligned} \tag{5.12}$$

Now compare (5.12) with (5.2), and it is easy to see that the only difference is that (5.2) uses $\|K - S\|_F$ as the regularizer of K while (5.12) uses $\|K - S\|_F^2$, which is a sufficient condition for the inner problem of (5.8) to have a closed-form solution. Therefore, a more accurate interpretation of (5.8) is that it finds the best-case kernel matrix K for classification rather than the worst-case K.

Two algorithms to solve the Indefinite SVM were proposed in [64]: one is based on the projected gradient method and the other is based on the analytic center cutting plane method. A third algorithm was proposed in [20], which reformulated (5.8) as a semi-infinite quadratically constrained linear program, and a fourth algorithm that applies Nesterov's smooth optimization method [72] was proposed in [106]. Several extensions of the Indefinite SVM, including those extensions to support vector regression (SVR) [90] and one-class SVMs [86], were discussed in [65].

5.1.2 Modifying the Test Similarities

We discussed in Section 4.1 how to handle test similarities for spectrum clip, flip and shift on the training similarity matrix. The same problem arises when we learn a kernel matrix

from an indefinite similarity matrix. As stated before, ideally, we would like to find a surrogate kernel function $K(x, x')$ for the similarity function $\psi(x, x')$. However, only $K^\star$, the surrogate of S in the PSD cone, is learned from training. Given a test sample x, estimating its label using its unmodified similarities to the training samples $s \in \mathbb{R}^n$, that is,

$$\hat{y} = \operatorname{sgn}\left((c^\star)^T s + b^\star\right),$$

ignores the fact that $c^\star$ and $b^\star$ are trained on $K^\star$ not on S.

Based on the same spirit of using $\|K - S\|_F$ to restrict the search for K to the vicinity of S, given s, $\psi(x, x)$, S and $K^\star$, I propose to find the appropriately modified test similarities $\tilde{s}$ by solving

$$\begin{aligned} \underset{r,t}{\text{minimize}} \quad & \left\| \begin{bmatrix} K^\star & r \\ r^T & t \end{bmatrix} - \begin{bmatrix} S & s \\ s^T & \psi(x,x) \end{bmatrix} \right\|_F \\ \text{subject to} \quad & \begin{bmatrix} K^\star & r \\ r^T & t \end{bmatrix} \succeq 0, \end{aligned} \tag{5.13}$$

with variables $r \in \mathbb{R}^n$ and $t \in \mathbb{R}$, in the hope that the optimal solution $r^\star$, which will be our $\tilde{s}$, is related to s in a way that is similar to how $K^\star$ is related to S. The test sample x is then classified as

$$\hat{y} = \operatorname{sgn}\left((c^\star)^T r^\star + b^\star\right).$$

The problem in (5.13) can be easily written as a convex conic program, but it can be further simplified. By applying Lemma 5.1 with its condition that r be in the range of $K^\star$ expressed as [13, Appendix A.5.5]

$$\left(I - K^\star (K^\star)^\dagger\right) r = 0,$$

we have

$$\begin{bmatrix} K^\star & r \\ r^T & t \end{bmatrix} \succeq 0 \;\Leftrightarrow\; \left(I - K^\star (K^\star)^\dagger\right) r = 0,\; r^T K^\star r - t \leq 0,$$

and as a result, we can reduce (5.13) to the following quadratically constrained quadratic

program (QCQP):

$$\begin{aligned} \underset{r,t}{\text{minimize}} \quad & 2\|r-s\|_2^2 + \left(t - \psi(x,x)\right)^2 \\ \text{subject to} \quad & r^T K^\star r - t \leq 0, \\ & \left(I - K^\star (K^\star)^\dagger\right) r = 0, \end{aligned} \tag{5.14}$$

which can be solved efficiently using interior point methods [34, 67, 71].

5.1.3 Formulation for Transductive Learning

Section 5.1.2 gives a QCQP formulation for modifying test similarities under the setting of inductive learning. I want to point out that for the case of transductive learning, where training and test samples are provided altogether, the task of training an SVM and that of learning kernel values for both training and test samples can be combined into one single optimization problem. Here we assume that besides n training samples, we also have n' test samples, and temporarily for the transductive learning setting, we are given an $(n+n') \times (n+n')$ indefinite similarity matrix S, which can be partitioned as follows,

$$S = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix},$$

where the upper left submatrix $S_{11} \in \mathbb{R}^{n \times n}$ is the similarity matrix for the n training samples. Assume the kernel matrix $K \in \mathbb{R}^{(n+n') \times (n+n')}$ that we would like to learn for both training and test samples has the same partition, that is,

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix},$$

where the upper left submatrix $K_{11} \in \mathbb{R}^{n \times n}$ is the kernel matrix for the n training samples, then we can form the following problem as a transductive learning extension of (5.2):

$$\begin{aligned} \underset{c,b,\xi,K}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T \xi + \eta c^T K_{11} c + \gamma \|K - S\|_F \\ \text{subject to} \quad & \operatorname{diag}(y)(K_{11}c + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & K \succeq 0. \end{aligned} \tag{5.15}$$

By the same technique that we used before, we can rewrite (5.15) as the following convex conic program:

$$\begin{aligned}
\underset{z,b,\xi,K,u,v}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta u + \gamma v \\
\text{subject to} \quad & \operatorname{diag}(y)(z + b\mathbf{1}) \geq \mathbf{1} - \xi, \\
& \xi \geq 0, \\
& \begin{bmatrix} K_{11} & z \\ z^T & u \end{bmatrix} \succeq 0, \\
& K \succeq 0, \\
& \|K - S\|_F \leq v,
\end{aligned} \tag{5.16}$$

which now has two LMI constraints. There is no need for modifying test similarities if we solve (5.16), because the optimal $K_{12}^\star$ already has the learned kernel values between the training and test samples, which should be used to classify the n' test samples.

Note that the main purpose of the above transductive learning extension of (5.2) is to avoid modifying test similarities separately — it is not an attempt to combine the concept of learning kernels from indefinite similarities with the transductive support vector machine (TSVM). Although it is possible to do so, it is out of the scope of this dissertation since TSVM itself is a big topic. We refer the reader to [19, Chapter 6 & 7] and [110, Chapter 6] for more details on the TSVM.

5.2 Learning the Spectrum Modification

Although (5.6) is a convex optimization problem, the scale of the problem, as measured by the number of (scalar) variables, grows quadratically with the number of training samples. Moreover, the flexibility of learning the whole kernel matrix K may lead to a model that overfits the data. Therefore, in this section, I propose a method that restricts K to be a spectrum modification of S and also has lower computational cost.

As discussed in Section 4.1, spectrum clip and flip can both be represented by matrix multiplications, that is, $\tilde{S} = PS$, where P is the transformation matrix. Recall that $S = U\Lambda U^T$, where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. For spectrum clip and flip, their corresponding

transformation matrices P_{clip} and P_{flip} share the same form:

$$P = U \operatorname{diag}(\alpha) U^T, \tag{5.17}$$

where $\alpha \in \mathbb{R}^n$, and we have

$$\alpha_{\text{clip}} = \begin{bmatrix} I_{\{\lambda_1 \geq 0\}} & \dots & I_{\{\lambda_n \geq 0\}} \end{bmatrix}^T,$$

and

$$\alpha_{\text{flip}} = \begin{bmatrix} \operatorname{sgn}(\lambda_1) & \dots & \operatorname{sgn}(\lambda_n) \end{bmatrix}^T.$$

Inspired by (5.17), I propose to restrict the surrogate kernel matrix K to be

$$K_\alpha = U \operatorname{diag}(\alpha) U^T S = U \operatorname{diag}(\alpha) U^T U \Lambda U^T = U \operatorname{diag}(\alpha) \Lambda U^T, \tag{5.18}$$

and treat α, which modifies the spectrum of S, as a variable to learn instead of the whole kernel matrix K. With the restriction given by (5.18), the number of (scalar) variables that we need to learn from the data grows linearly with the number of training samples instead of quadratically, and the model is less flexible and thus less prone to overfitting. A third benefit is that we no longer need to solve the QCQP given by (5.14) for each test sample; after we learn the spectrum modification vector α, for a test sample x with similarity vector s, as proposed in Section 4.1, we can modify its similarities to the training samples using the same matrix multiplication:

$$\tilde{s} = Ps = U \operatorname{diag}(\alpha) U^T s. \tag{5.19}$$

Using (5.18), I propose the following problem to jointly learn a spectrum modification and train an SVM:

$$\begin{aligned} \underset{c, b, \xi, \alpha}{\text{minimize}} \quad & \frac{1}{n} \mathbf{1}^T \xi + \eta c^T K_\alpha c + \gamma h(\alpha) \\ \text{subject to} \quad & \operatorname{diag}(y)(K_\alpha c + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & \Lambda \alpha \geq 0, \end{aligned} \tag{5.20}$$

where $h(\alpha)$ is a convex regularizer of α. From now on, I will refer to (5.20) as the *SimSVM*. As in Section 5.1, by introducing slack variables $t, v \in \mathbb{R}$ and letting $z = K_\alpha c$, we can

rewrite (5.20) as the following convex optimization problem:

$$\begin{aligned} \underset{z,b,\xi,\alpha,t,v}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta t + \gamma v \\ \text{subject to} \quad & \operatorname{diag}(y)(z + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & \begin{bmatrix} K_\alpha & z \\ z^T & t \end{bmatrix} \succeq 0, \\ & h(\alpha) \leq v. \end{aligned} \tag{5.21}$$

The type of the last constraint $h(\alpha) \leq v$ depends on what kind of $h(\alpha)$ we choose as the regularizer, which will be discussed below.

5.2.1 Options for the Regularizer $h(\alpha)$

In Section (5.1), we regularize the search for K toward S using $\|K - S\|_F$. Since

$$\|K_\alpha - S\|_F = \|U(\operatorname{diag}(\alpha) - I)\Lambda U^T\|_F = \|(\operatorname{diag}(\alpha) - I)\Lambda\|_F = \|\Lambda(\alpha - \mathbf{1})\|_2,$$

$\|\Lambda(\alpha - \mathbf{1})\|_2$ could be a reasonable option for $h(\alpha)$. As mentioned in Section (5.1), when the regularization parameter γ is set large, using $\|K - S\|_F$ makes (5.2) very close to training an SVM with S_{clip}, and hence we expect that using $h(\alpha) = \left\|\alpha - \alpha_{\text{clip}}\right\|_2$ will achieve similar results as using $h(\alpha) = \|\Lambda(\alpha - \mathbf{1})\|_2$, which we have verified experimentally. We have also observed that for certain S that has many near-zero eigenvalues, using $h(\alpha) = \left\|\alpha - \alpha_{\text{clip}}\right\|_2$ can make some convex optimization solvers less prone to numerical instability.

In fact, instead of searching in the vicinity of S, one might want to regularize the search for K_α toward other approximations of S. For example, one can use other regularizers such as $h(\alpha) = \left\|\alpha - \alpha_{\text{flip}}\right\|_2$, and we have seen from the results in Table 4.6 that spectrum flip helps both SVM and SVM-KNN achieve their best results on the Protein data set.

Choosing either $h(\alpha) = \left\|\alpha - \alpha_{\text{clip}}\right\|_2$ or $h(\alpha) = \left\|\alpha - \alpha_{\text{flip}}\right\|_2$ will make $h(\alpha) \leq v$ an SOC constraint. However, if we change the ℓ_2-norm to the ℓ_1-norm, that is, we let $h(\alpha) = \left\|\alpha - \alpha_{\text{clip}}\right\|_1$ or $h(\alpha) = \left\|\alpha - \alpha_{\text{flip}}\right\|_1$, then $h(\alpha) \leq v$ can be expressed as a set of linear constraints. Moreover, due to the fact that the ℓ_1-norm encourages sparse solutions [13], one

should expect that many components of α will be exactly the same as those of α_{clip} or α_{flip}, depending on which one is used in the regularizer.

5.2.2 *Linear Program Approximation*

Being able to formulate the SimSVM as a convex optimization problem does not mean that we are able to solve it fast enough for practical uses; even though the number of (scalar) variables in (5.21) grows linearly with n, due to its large LMI constraint, it is still impractical to solve (5.21) on large or even medium size data sets. In this subsection, we describe an attempt for a fast algorithm by using an LP to approximate (5.21). The rationale of using an LP to approximate the original problem is that LPs are generally considered efficiently solvable in both theory and practice, and modern commercial solvers can handle LPs with millions of variables and constraints. Another attempt that reformulates (5.20) as a second-order cone program (SOCP) will be described in the next subsection.

The key idea of the LP approximation is to replace the LMI constraint in (5.21) with the following equality constraint:

$$\begin{bmatrix} K_\alpha & z \\ z^T & t \end{bmatrix} = \sum_{i=1}^{m} w_i \beta_i \beta_i^T, \tag{5.22}$$

for some $\beta_i \in \mathbb{R}^{n+1}$, $i = 1, \ldots, m$, where $w_i \geq 0$, $i = 1, \ldots, m$. That is, we approximate the LMI in (5.21) by expressing the matrix on the left side of the LMI as a conic combination of m rank-one PSD matrices constructed from a set of m preselected vectors $\{\beta_i\}_{i=1}^m$. This approximation is adapted from the idea proposed in [51]. The authors of [51] proposed an algorithm termed *random conic pursuit* to solve semidefinite programs (SDPs) [100] via repeated optimization over randomly selected two-dimensional subcones of the PSD cone. For the ith iteration, they construct the following subcone:

$$\mathcal{C}_i = \left\{ X \in \mathbb{R}^{n \times n} \;\middle|\; X = aX_{i-1} + bx_i x_i^T,\ a, b \geq 0 \right\},$$

where $X_{i-1} \in \mathbb{R}^{n \times n}$ is the solution from the previous iteration, and $x_i \in \mathbb{R}^n$ is a random vector sampled from a multivariate Gaussian distribution. Then they optimize over $\mathcal{C}_i$ by using a double bisection search for variables a and b. With an initial solution $X_0 \succeq 0$, their

solution at the ith iteration can be expressed as

$$X_i = w_0 X_0 + \sum_{j=1}^{i} w_j x_j x_j^T, \tag{5.23}$$

for some $w_j \geq 0$, $j = 0, \ldots, i$. The approximation given by (5.22) was inspired by random conic pursuit in that both construct the final PSD matrix solution as a conic combination of rank-one PSD matrices. However, because of the special structure of the matrix we have in (5.22), we can be more efficient than choosing the basis matrices completely at random.

It is clear that the conic combination of the m rank-one PSD matrices can only represent a subset of all $(n+1) \times (n+1)$ PSD matrices, but we hope that by choosing a large m and carefully designing the set $\{\beta_i\}_{i=1}^m$, we can obtain a decent approximation to the original optimal solution. To construct the rank-one PSD matrices in (5.22), first, for each $i \in \{1, \ldots, n\}$, we randomly draw p i.i.d. samples τ_{ij}, $j = 1, \ldots, p$, from some probability distribution on $\mathbb{R}$. Then, we let

$$\beta_{ij} = \begin{bmatrix} u_i \\ \tau_{ij} \end{bmatrix},$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, where u_i denotes the ith column of U. With these β_{ij}'s, the approximation is given by

$$\begin{bmatrix} K_\alpha & z \\ z^T & t \end{bmatrix} = \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \beta_{ij} \beta_{ij}^T = \begin{bmatrix} \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} G_i & \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \tau_{ij} u_i \\ \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \tau_{ij} u_i^T & \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \tau_{ij}^2 \end{bmatrix}, \tag{5.24}$$

where $G_i = u_i u_i^T$, $i = 1, \ldots, n$. Since

$$K_\alpha = U \operatorname{diag}(\alpha) \Lambda U^T = \sum_{i=1}^{n} \lambda_i \alpha_i G_i,$$

where α_i is the ith component of α, and

$$\langle G_i, G_j \rangle = \operatorname{tr}(G_i^T G_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases}$$

under the approximation given by (5.24), we have

$$\lambda_i \alpha_i = \sum_{j=1}^{p} w_{ij}, \tag{5.25}$$

for $i = 1, \ldots, n$. From (5.24), we also have

$$z = \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \tau_{ij} u_i, \tag{5.26}$$

and

$$t = \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \tau_{ij}^2. \tag{5.27}$$

Given the relationships in (5.25)–(5.27), and assume, for example, $h(\alpha) = \left\| \alpha - \alpha_{\text{clip}} \right\|_1$, we can then solve (5.21), which, in this case, is an SDP, approximately by solving the following LP:

$$\begin{aligned} \underset{w, b, \xi, v}{\text{minimize}} \quad & \frac{1}{n} \mathbf{1}^T \xi + \eta \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \tau_{ij}^2 + \gamma \mathbf{1}^T v \\ \text{subject to} \quad & \operatorname{diag}(y) \left(\sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} \tau_{ij} u_i + b\mathbf{1} \right) \geq \mathbf{1} - \xi, \\ & -v_i \leq \frac{1}{\lambda_i} \sum_{j=1}^{p} w_{ij} - \left(\alpha_{\text{clip}} \right)_i \leq v_i, \quad i = 1, \ldots, n, \\ & \xi \geq 0, \quad w \geq 0, \quad v \geq 0, \end{aligned} \tag{5.28}$$

with variables $w \in \mathbb{R}^{np}$, $b \in \mathbb{R}$, $\xi \in \mathbb{R}^n$ and $v \in \mathbb{R}^n$. Notice that in (5.28), we assume $\lambda_i \neq 0$, $i = 1, \ldots, n$; if there exist λ_i's that are zero, we can simply remove their corresponding terms in (5.28) and only solve for those w_{ij}'s that correspond to nonzero λ_i's.

5.2.3 Second-Order Cone Program Formulation

In this subsection, we describe a reformulation of (5.20) that takes advantage of the special structure of learning the spectrum modification and results in a convex optimization problem that is easier to solve than (5.21).

First, we let $\tilde{c} = U^T c$, and given (5.18), we can rewrite (5.20) as

$$\begin{aligned} \underset{\tilde{c}, b, \xi, \alpha}{\text{minimize}} \quad & \frac{1}{n} \mathbf{1}^T \xi + \eta \tilde{c}^T \operatorname{diag}(\alpha) \Lambda \tilde{c} + \gamma h(\alpha) \\ \text{subject to} \quad & \operatorname{diag}(y)(U \operatorname{diag}(\alpha) \Lambda \tilde{c} + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \\ & \Lambda \alpha \geq 0. \end{aligned} \tag{5.29}$$

Then we let $z = \operatorname{diag}(\alpha)\Lambda\tilde{c}$, that is, $z_i = \lambda_i\alpha_i\tilde{c}_i$, $i = 1, \ldots, n$. Because

$$\tilde{c}^T \operatorname{diag}(\alpha)\Lambda\tilde{c} = \sum_{i=1}^{n} \lambda_i\alpha_i\tilde{c}_i^2,$$

and given $\lambda_i\alpha_i \geq 0$, $t_i \geq 0$, $i = 1, \ldots, n$, we have

$$\lambda_i\alpha_i\tilde{c}_i^2 \leq t_i \;\Leftrightarrow\; \lambda_i^2\alpha_i^2\tilde{c}_i^2 \leq \lambda_i\alpha_i t_i \;\Leftrightarrow\; z_i^2 \leq \lambda_i\alpha_i t_i.$$

Thus, we can rewrite (5.29) as

$$\begin{aligned} \underset{z,\,b,\,\xi,\,\alpha,\,t}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta\mathbf{1}^T t + \gamma h(\alpha) \\ \text{subject to} \quad & \operatorname{diag}(y)(Uz + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \quad t \geq 0, \\ & \Lambda\alpha \geq 0, \\ & z_i^2 \leq \lambda_i\alpha_i t_i, \quad i = 1, \ldots, n, \end{aligned} \tag{5.30}$$

where $t \in \mathbb{R}^n$ is the vector of n slack variables. What stands out in (5.30) is the set of hyperbolic constraints on the last line. Before we proceed, we need to introduce the following lemma that establishes the equivalence between restricted hyperbolic constraints and SOC constraints.

Lemma 5.3. *For $x \in \mathbb{R}^n$ and $y, z \in \mathbb{R}$, we have*

$$x^T x \leq yz, \quad y \geq 0, \quad z \geq 0,$$

if and only if

$$\left\| \begin{bmatrix} 2x \\ y - z \end{bmatrix} \right\|_2 \leq y + z, \quad y \geq 0, \quad z \geq 0.$$

Lemma 5.3 is well-known in the convex optimization community [13, 1, 62]. For completeness, I provide the proof and a geometric interpretation in Appendix B.

By applying Lemma 5.3 to those hyperbolic constraints, we have

$$z_i^2 \leq \lambda_i\alpha_i t_i,\ \lambda_i\alpha_i \geq 0,\ t_i \geq 0 \;\Leftrightarrow\; \left\| \begin{bmatrix} 2z_i \\ \lambda_i\alpha_i - t_i \end{bmatrix} \right\|_2 \leq \lambda_i\alpha_i + t_i,\ \lambda_i\alpha_i \geq 0,\ t_i \geq 0,$$

for $i = 1, \ldots, n$. Given the above equivalence and assume, for example, we use the regularizer $h(\alpha) = \left\|\alpha - \alpha_{\text{clip}}\right\|_2$, we can rewrite (5.30) as

$$\begin{aligned}
\underset{z,b,\xi,\alpha,t,v}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta\mathbf{1}^T t + \gamma v \\
\text{subject to} \quad & \operatorname{diag}(y)(Uz + b\mathbf{1}) \geq \mathbf{1} - \xi, \\
& \xi \geq 0, \quad t \geq 0, \\
& \Lambda\alpha \geq 0, \\
& \left\| \begin{bmatrix} 2z_i \\ \lambda_i\alpha_i - t_i \end{bmatrix} \right\|_2 \leq \lambda_i\alpha_i + t_i, \quad i = 1, \ldots, n, \\
& \left\|\alpha - \alpha_{\text{clip}}\right\|_2 \leq v,
\end{aligned} \tag{5.31}$$

with variables $z, \xi, \alpha, t \in \mathbb{R}^n$ and $b, v \in \mathbb{R}$. The problem in (5.31) is a second-order cone program (SOCP) [1, 62], and can be efficiently solved by algorithms such as the primal-dual interior point method [3]. Compare (5.31) with (5.21), and we can find that the large LMI constraint in (5.21) is now replaced in (5.31) by n SOC constraints , each of which involves only a two-dimensional vector. This key difference makes (5.31) much easier to solve than (5.21). Also notice that unlike the LP approximation discussed in the previous subsection, (5.31) is equivalent to (5.20), and therefore solving (5.31) gives the exact optimal solution to the original problem.

Lastly, let $z^\star$ and $\alpha^\star$ denote the optimal solution to (5.31), and since we let

$$z = \operatorname{diag}(\alpha)\Lambda\tilde{c} = \operatorname{diag}(\alpha)\Lambda U^T c,$$

we can recover the optimal $c^\star$ by

$$c^\star = U\left(\operatorname{diag}(\alpha^\star)\Lambda\right)^\dagger z^\star.$$

5.3 Experiments

In this section, we first compare the method using the LP approximation to solve the SimSVM as detailed in Section 5.2.2 and the method using the SOCP formulation as described in Section 5.2.3. Then we compare the proposed SimSVM with the SVMs using

the similarities as a kernel via spectrum clip, flip and shift, and the Indefinite SVM given by (5.8), which, as shown in Section 5.1.1, is equivalent to learning the full kernel matrix.

5.3.1 LP Approximation vs SOCP Formulation

We tested the LP approximation method and the SOCP method on the Aural Sonar and Voting data sets, which are described in Section 4.2. Both data sets have two classes. To be able to apply the LP approximation, we chose the regularizer $h(\alpha) = \|\alpha - \alpha_{\text{clip}}\|_1$ for both methods, and the two regularization parameters were set to $\eta = \gamma = 0.01$. The convex optimization solver MOSEK[1] was used to solve both the LP and the SOCP. MOSEK offers four methods for solving the LP: the primal simplex method, the dual simplex method, the primal-dual simplex method, and the interior point method. We used the dual simplex method for it was the fastest when tested on our problems. The computer on which we ran these experiments has an Intel Core i7 3.2 GHz processor and 12 GB memory and runs the Linux operating system.

In the LP approximation method, for each $i \in \{1, \ldots, n\}$, we generated τ_{ij}, $j = 1, \ldots, p$, from a Gaussian distribution whose mean and variance are equal to the sample mean and variance of the elements of u_i, respectively. We tested four different choices of p: 10, 50, 100 and 200.

The goal of this set of experiments is to answer the following two questions:

1. Can the LP approximation method yield solutions that are reasonably close to those obtained using the SOCP method?

2. Can we gain a shorter running time by using the LP approximation method?

Unfortunately, the results unveiled in Table 5.1 and 5.2 show that the performance of the LP approximation method is quite disappointing. Table 5.1 shows the running time (in milliseconds), the optimal value of the objective function, the empirical risk defined as the average hinge loss on the training set, and the empirical classification error defined as the

[1] Available at `http://www.mosek.com/`

percent of training samples misclassified. Table 5.2 shows the relative differences between the LP approximation method and the SOCP method, which is used as the baseline, in terms of the optimal value of the objective function and the optimal solutions of the SVM parameters c and b and the spectrum modification vector α. Here the relative difference between a scalar or vector x and its baseline x_0 is defined as

$$E_x = \frac{\|x - x_0\|_2}{\|x_0\|_2} \times 100\%.$$

Because the directly measured running time is not a constant and the LP approximation method uses random numbers, the results shown in Table 5.1 and 5.2 are averaged over 10 runs.

It is easy to see that as p, the number of random samples drawn for each i, goes larger, the results of the LP approximation method get closer to those of the SOCP method, especially the empirical risk and classification error, which are most pertinent to our classification tasks. This is as expected since with a larger p, we have more rank-one PSD matrices for the approximation given by (5.22). However, as can be seen from Table 5.2, even when $p = 200$, there are still significant relative differences in the SVM parameters c and b and the spectrum modification vector α. The most disappointing part of the LP approximation method is that even when $p = 10$, it already takes longer to solve the LP given by (5.28) than the SOCP given by (5.31) as shown in Table 5.1, not to mention when p grows larger. Notice that the running time of both methods on the Voting data set is less than that on the Aural Sonar data set even though the Voting data set has 435 samples while the Aural Sonar data set has only 100 samples. This is because the similarity matrix of the Voting data set has many near-zero eigenvalues and the decision variables corresponding to these near-zero eigenvalues are removed from the optimization problems.

In conclusion, compared with the SOCP method for solving the SimSVM, the LP approximation method is not very effective in terms of both accuracy and speed. Therefore, for the experiments on classification performance, which will be discussed in the next subsection, we used the SOCP method for implementing the SimSVM.

Table 5.1: Performances of the LP approximation method and the SOCP method are compared below in terms of the running time (in millisecond), optimal value of the objective function, and risk and classification error (in percentage) on the training set. The results are averaged over 10 runs, and their standard deviations are shown in the parentheses.

		p	Aural Sonar		Voting	
Running Time (ms)	SOCP		44.6	(2.7)	38.3	(3.4)
	LP	10	134.3	(7.1)	56.4	(1.5)
	LP	50	550.2	(22.3)	367.1	(8.5)
	LP	100	1093.5	(68.4)	740.5	(31.2)
	LP	200	2479.2	(214.0)	1354.8	(19.6)
Optimal Value	SOCP		0.283		0.185	
	LP	10	0.366	(0.047)	0.255	(0.033)
	LP	50	0.303	(0.004)	0.219	(0.006)
	LP	100	0.301	(0.004)	0.217	(0.008)
	LP	200	0.294	(0.002)	0.211	(0.005)
Empirical Risk	SOCP		0.166		0.102	
	LP	10	0.241	(0.028)	0.177	(0.018)
	LP	50	0.201	(0.002)	0.151	(0.008)
	LP	100	0.198	(0.003)	0.149	(0.011)
	LP	200	0.192	(0.001)	0.140	(0.013)
Empirical Error (%)	SOCP		6.0		4.4	
	LP	10	10.3	(1.5)	7.0	(1.2)
	LP	50	7.8	(0.8)	5.6	(0.4)
	LP	100	7.4	(0.5)	5.6	(0.6)
	LP	200	7.0	(0.0)	5.2	(0.5)

Table 5.2: Percent difference between the LP approximation method and the SOCP method (used as the baseline) in terms of the optimal value of the objective function f and the optimal solutions of $c \in \mathbb{R}^n$, $b \in \mathbb{R}$ and $\alpha \in \mathbb{R}^n$. The results are averaged over 10 runs, and their standard deviations are shown in the parentheses.

	p	Aural Sonar		Voting	
E_f (%)	10	29.5	(16.6)	37.8	(18.0)
	50	7.0	(1.3)	18.0	(3.3)
	100	6.3	(1.5)	17.0	(4.4)
	200	3.9	(0.6)	13.7	(2.4)
E_c (%)	10	68.4	(5.7)	79.8	(5.4)
	50	49.4	(2.2)	67.5	(5.7)
	100	46.2	(1.8)	65.1	(5.8)
	200	40.0	(1.9)	59.5	(6.5)
E_b (%)	10	661.1	(1032.0)	47.4	(44.0)
	50	27.8	(15.4)	72.0	(22.0)
	100	31.1	(15.3)	59.9	(26.7)
	200	24.7	(13.0)	65.0	(16.6)
E_α (%)	10	64.1	(22.0)	68.6	(69.0)
	50	21.9	(6.8)	32.2	(7.6)
	100	20.4	(8.1)	29.1	(7.7)
	200	11.9	(5.5)	28.1	(4.9)

5.3.2 *Classification Test*

In this subsection, we compare five SVM methods that treat similarities as a kernel: SVMs with spectrum clip, flip and shift, the Indefinite SVM given by (5.8), which learns the full kernel matrix, and the SimSVM proposed in Section 5.2, which learns the spectrum modification. For the Indefinite SVM, we used the code provided by the authors of [65], which solves the Indefinite SVM using both the projected gradient method and the analytic center cutting plane method. For our experiments, we used the projected gradient method, which, according to the experiments reported in [65] and some of our own experiments, is more efficient than the analytic center cutting plane method. For the SimSVM, we used MOSEK to solve the SOCP formulation given by (5.31).

We ran the experiments on eight binary classification tasks using eight real data sets representing a diverse set of indefinite similarities. These eight data sets are: Amazon-2, Aural Sonar, Protein-2, Voting, Yeast-5-7, Yeast-5-12, Yeast-6-8 and Yeast-6-10. Among them, the Aural Sonar and Voting data sets are already described in Section 4.2.

The *Amazon-2* data set is a subset of the Amazon-47 data set described in Section 4.2, and therefore has the same type of similarities. We manually labeled 96 books by 23 different authors as either fiction or nonfiction, and the problem is to correctly classify each book as one of the 36 nonfiction books or one of the 60 fiction books.

The *Protein-2* data set is a subset of the Protein data set described in Section 4.2. The original data set has 213 samples and 4 classes. In order to use it for binary classification, we treat the problem as classifying the two most confusable classes, each of which has 72 samples.

The *Yeast-5-7*, *Yeast-5-12*, *Yeast-6-8* and *Yeast-6-10* data sets are all subsets of the Yeast data set used in [56]. The problem is to predict the functions of yeast proteins. The original Yeast data set contains 3588 samples and each sample is a yeast protein sequence, and the Smith-Waterman E-value is used to measure the similarity between two protein sequences. There are 13 classes in the original data set and some samples belong to more than one class due to their multiple roles. To simplify the problem and adapt it to binary classification, we arbitrarily chose four pairs of classes. Each subset with the name "Yeast-A-B" was formed

by selecting samples that belong to either class A or class B, and then eliminating those that belong to both classes.

We symmetrized all the similarity matrices and normalized their entries to the range of $[0, 1]$. For each data set, we randomly partitioned the data 20 times into 20% test and 80% training. For each of the 20 partitions, we selected parameters by a 10-fold cross-validation on the training set. The hyperparameter C for the traditional C-SVM and the Indefinite SVM, the regularization parameter ρ for the Indefinite SVM, and the regularization parameters η and γ for the SimSVM were cross-validated from the following sets:

$$
\begin{aligned}
C &\in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}, \\
\rho &\in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}, \\
\eta &\in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}, \\
\gamma &\in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\},
\end{aligned}
$$

except that for the C-SVM with spectrum shift, its hyperparameter C was cross-validated from

$$C \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\},$$

because we observed from preliminary experiments that the SVM with spectrum shift tends to favor small values for C more often than the SVMs with spectrum clip and flip. For the SimSVM, we used the regularizer $h(\alpha) = \left\|\alpha - \alpha_{\text{clip}}\right\|_2$ for all the data sets except that for the Protein-2 data set, we used $h(\alpha) = \left\|\alpha - \alpha_{\text{flip}}\right\|_2$ based on our prior knowledge gained from the comparative study discussed in Chapter 4 that this particular data set favors spectrum flip.

The test errors averaged over the 20 randomized test/training partitions along with the standard deviations (in parentheses) are shown in Table 5.3. For each data set, the bold results denote the classifier with the lowest average error and those not statistically significantly worse according to a one-sided Wilcoxon signed-rank test at a significance level of 5%. Due to the excessive time it takes to run the Indefinite SVM,[2] we were only

[2]For example, on a computer that has an AMD Athlon X2 2.6 GHz processor and 2 GB memory and runs the Windows XP operating system, it took the Indefinite SVM about five days to finish the 20 runs on the Voting data set.

Table 5.3: Mean and standard deviation (in parentheses) of the test errors (in percentage) across 20 randomized test/training partitions. For each data set, the lowest mean error and those not statistically significantly worse are boldfaced.

	Amazon-2	Aural Sonar	Protein-2	Voting
# Samples	96	100	144	435
SimSVM	11.05 (7.99)	**12.00** (4.97)	**3.10** (3.34)	**4.60** (2.04)
Indefinite SVM	**7.63** (6.72)	**12.00** (5.94)	18.28 (5.72)	**5.11** (1.88)
SVM w/ clip	11.84 (7.80)	**13.00** (5.48)	8.79 (5.06)	**4.89** (2.11)
SVM w/ flip	20.00 (11.66)	13.25 (5.45)	4.83 (3.04)	**4.94** (2.08)
SVM w/ shift	17.89 (12.00)	32.75 (18.95)	26.55 (7.68)	**5.00** (1.76)
	Yeast-5-7	Yeast-5-12	Yeast-6-8	Yeast-6-10
# Samples	814	635	778	909
SimSVM	**18.31** (3.10)	**8.74** (2.46)	**23.75** (3.00)	**29.40** (3.19)
Indefinite SVM	N/A	N/A	N/A	N/A
SVM w/ clip	19.75 (3.52)	9.53 (2.41)	24.84 (2.98)	**29.89** (3.07)
SVM w/ flip	22.15 (2.94)	10.51 (2.67)	25.74 (3.56)	31.18 (3.69)
SVM w/ shift	43.28 (5.59)	34.39 (8.34)	28.75 (2.71)	41.04 (3.17)

able to provide the results of the Indefinite SVM on the four smaller data sets as shown in Table 5.3. Truly, the computational cost of the SimSVM is also higher than that of the traditional SVM, but it still takes much less time to run the SimSVM than the Indefinite SVM, as we observed from our experiments.

One can see from Table 5.3 that the proposed SimSVM is among the top performers on seven out of the eight data sets. This indicates that on some indefinite similarity data sets, the SimSVM that learns the spectrum modification can achieve statistically significant improvement over the SVMs with simple spectrum modification such as clip, flip and shift. On the Amazon-2 data set, which is the only sparse data set here, the Indefinite SVM that learns the full kernel matrix is obviously the winner. To investigate this, we trained the SimSVM and the Indefinite SVM on all the samples of the Amazon-2 data set using the

parameters selected most frequently over the 20 random partitions. The similarity matrix S and the kernel matrix K_α learned by the SimSVM are shown in Figure 5.1(a) and (b), respectively. It is easy to see that the learned spectrum modification does not change the sparseness of the similarity matrix. This is further verified by the difference matrix shown in Figure 5.1(c), where each entry is the absolute difference between the corresponding entries of S and K_α. Let K denote the kernel matrix learned by the Indefinite SVM. To compare the SimSVM with the Indefinite SVM, we plot the difference matrix between K and K_α in Figure 5.1(d). The block structure revealed in this difference matrix is clearly a result of the rank-one update given by (5.9). Most of the entries in this difference matrix are in the order of 10^{-5} to 10^{-4}. Given that the original similarities are in the range of $[0, 1]$, and the range of the entries in K and K_α is close to $[0, 1]$, such differences seem to be very small, but we conjecture that these small differences introduced by the rank-one update still help make the kernel matrix less sparse and to certain degree uncover some hidden relationships between samples.

In addition to the classification results, we illustrate the behavior of the SimSVM in Figure 5.2–5.9. For each data set, we trained the SimSVM on all the samples of that data set using the parameters selected most frequently over the 20 random partitions. Then we plotted for each data set the similarity matrix S, the kernel matrix K_α, the spectrum modification vector α, and the spectrum before and after the modification. It is worth pointing out that the negative eigenvalues of the similarity matrix of the Voting data set are so small that they are hard to see in Figure 5.5(c). This explains why for the classification task on this data set, there is no statistically significant difference between the five SVM methods, as one can see from Table 5.3.

Our experiments also verify the necessity of treating training and test similarities consistently.[3] As an example, for the SimSVM, besides the results reported in Table 5.3 using the modified test similarities $\tilde{s}$ given by (5.19), we also recorded its test errors using unmodified test similarities s. Table 5.4 shows these two sets of test errors, and it is easy to

[3]This includes the Indefinite SVM. The code of the Indefinite SVM tries to modify training and test similarities consistently by using a heuristic based on (5.9), which is detailed in [65]. This heuristic seems to work well: for example, on the Protein-2 data set, if this heuristic were not used, the error rate of the Indefinite SVM would increase from 18.28% to 44.31%.

Table 5.4: Test errors (in percentage) averaged over 20 randomized test/training partitions.

	Amazon-2	Aural Sonar	Protein-2	Voting
SimSVM using $\tilde{s}$	11.05	12.00	3.10	4.60
SimSVM using s	13.68	14.50	44.31	5.40
	Yeast-5-7	Yeast-5-12	Yeast-6-8	Yeast-6-10
SimSVM using $\tilde{s}$	18.31	8.74	23.75	29.40
SimSVM using s	37.67	34.33	26.76	38.98

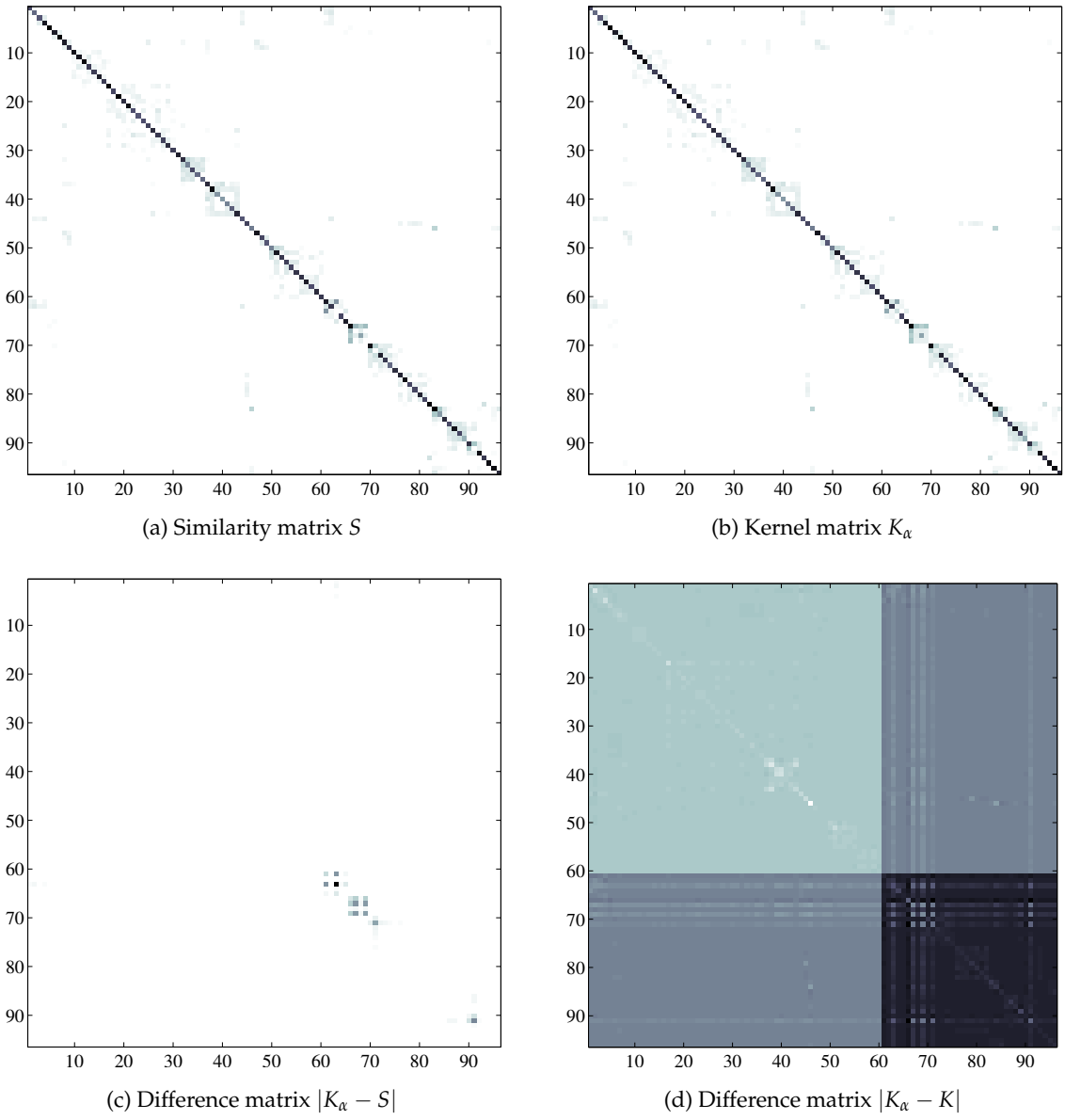
conclude that modifying test similarities by (5.19) is an indispensable step for the SimSVM.

5.4 Combining Similarities with Multiple Kernels

In some applications, there may be multiple possible descriptions of the similarities between data samples. An example of multiple similarity descriptions arises in the problem of protein classification in computational biology [56, 54]. Pairwise similarities between proteins can be based on protein-protein interactions, genetic interactions, co-participation in a protein complex, Smith-Waterman sequence matching algorithm [89], and other factors. In general, fusing multiple similarities with regard to a particular classification task can provide a task-specific view of the relationships between samples, and the performance may be better than with any single description.

A special case is when each similarity satisfies the mathematical properties of a kernel. In that case, one can treat each similarity as a kernel, and fuse the similarities using multiple kernel learning (MKL), in which a linear combination of multiple kernels and the parameters of a discriminative classifier acting on the kernel combination are jointly learned [55]. MKL can be used to fuse heterogeneous descriptions of data samples in the form of multiple kernels. For the above example of protein function prediction, it is shown in [56] and [54] that a classifier trained on a conic combination of all the given kernels yielded better classification results than the same classifier trained on any single kernel.

However, as we have seen, similarities can be indefinite and thus fail to satisfy the



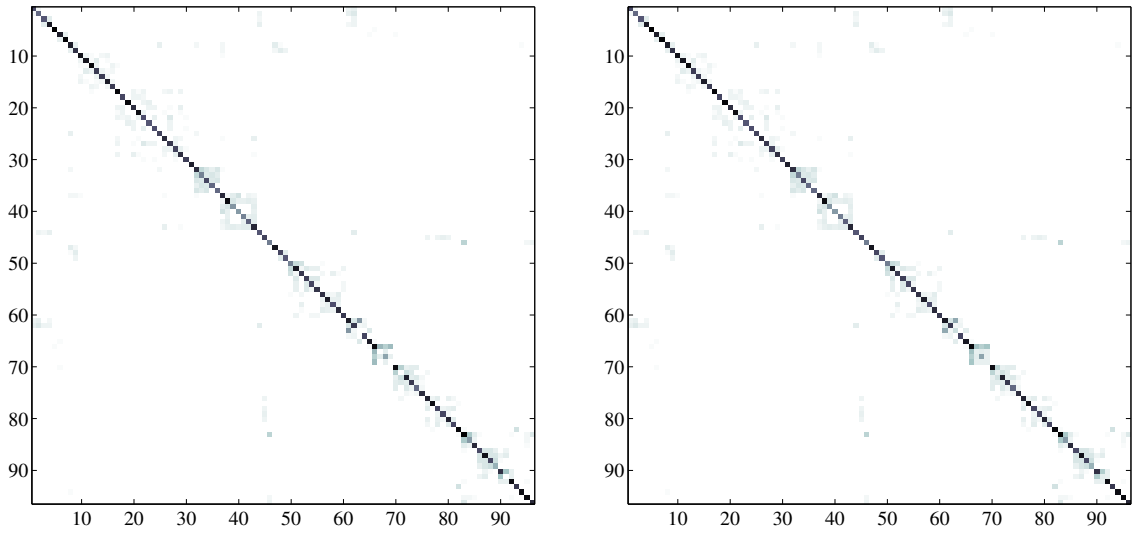
(a) Similarity matrix S

(b) Kernel matrix K_α

(c) Difference matrix $|K_\alpha - S|$

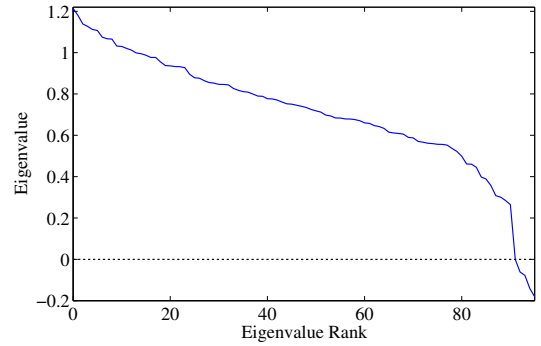
(d) Difference matrix $|K_\alpha - K|$

Figure 5.1: The similarity matrix S of the Amazon-2 data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. For each (i, j)-entry, the absolute difference between K_α and S, and that between K_α and K, which is the kernel matrix learned by the Indefinite SVM, are shown in (c) and (d), respectively. The darker the color is, the larger the corresponding value is; however, depending on the range of the data for each plot, the same color does not necessarily represent the same value across plots.

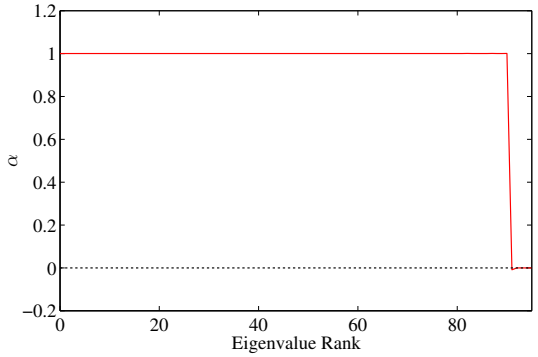


(a) Similarity matrix S

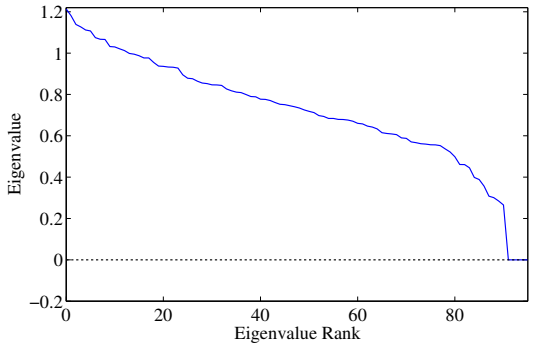
(b) Kernel matrix K_α



(c) Spectrum of S

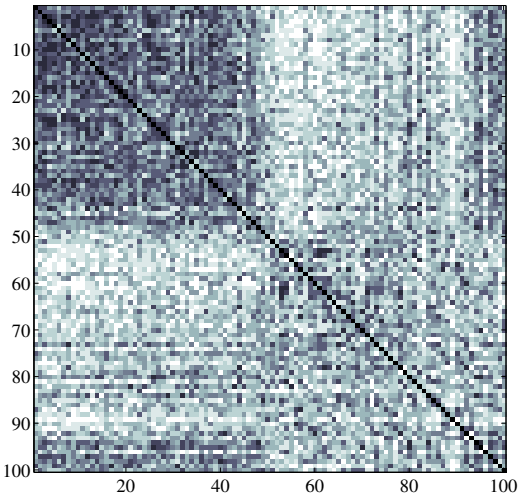


(d) Spectrum modification vector α

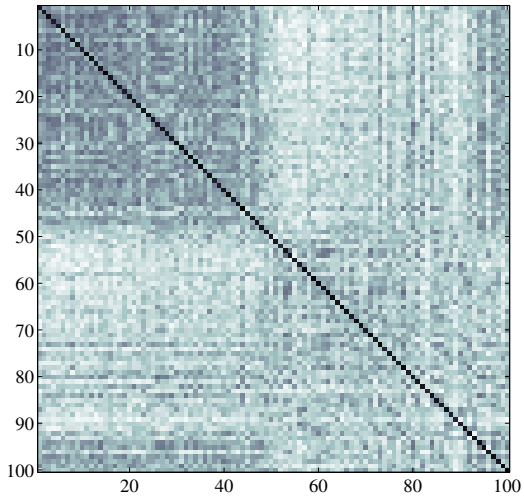


(e) Spectrum of K_α

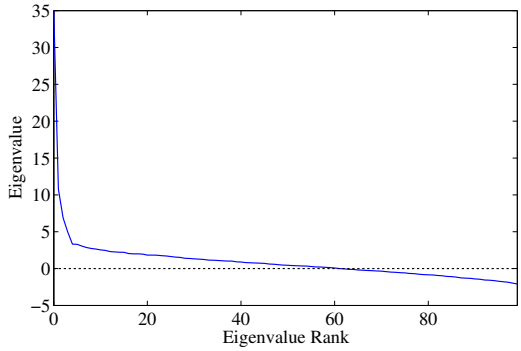
Figure 5.2: The similarity matrix S of the Amazon-2 data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.



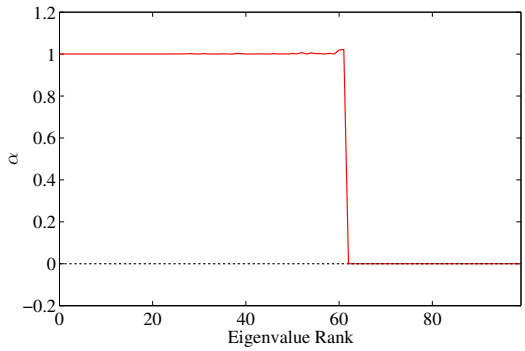
(a) Similarity matrix S



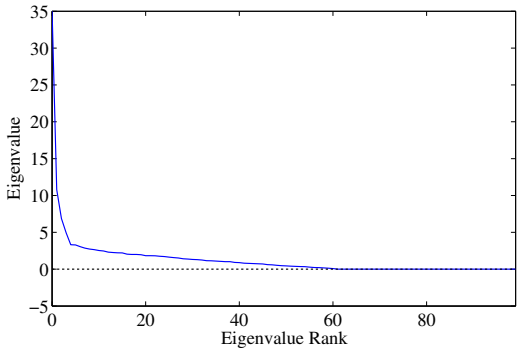
(b) Kernel matrix K_α



(c) Spectrum of S

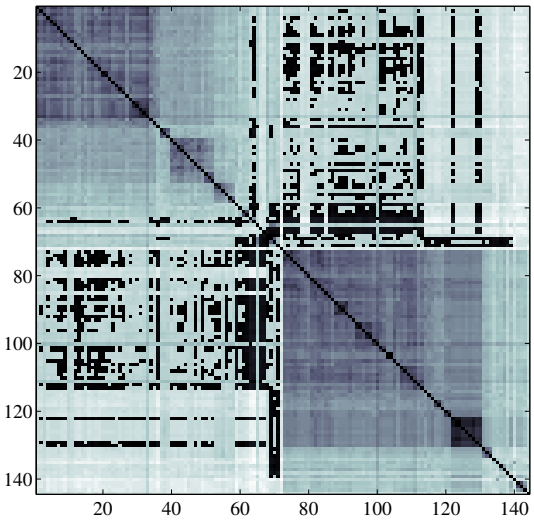


(d) Spectrum modification vector α

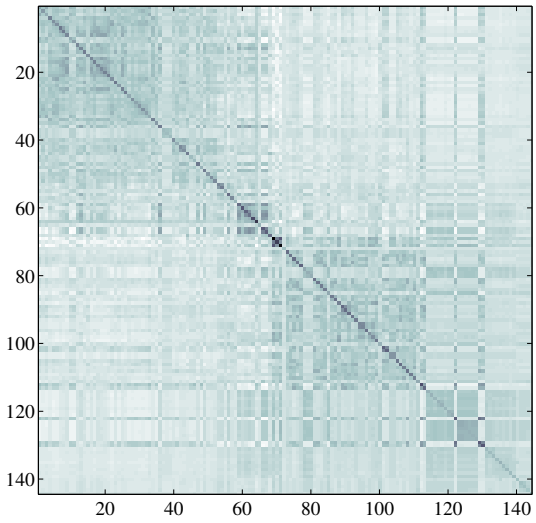


(e) Spectrum of K_α

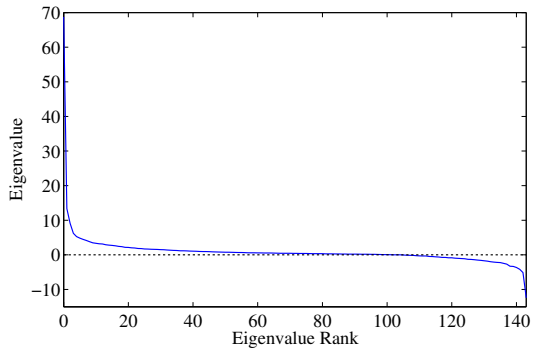
Figure 5.3: The similarity matrix S of the Aural Sonar data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.



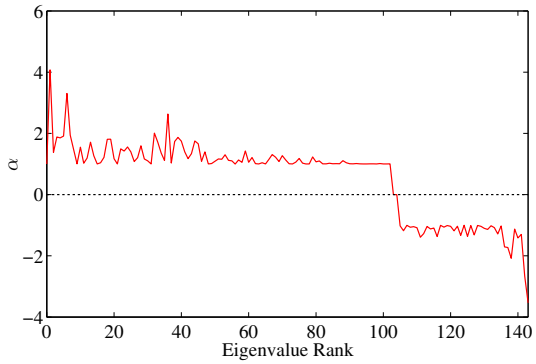
(a) Similarity matrix S



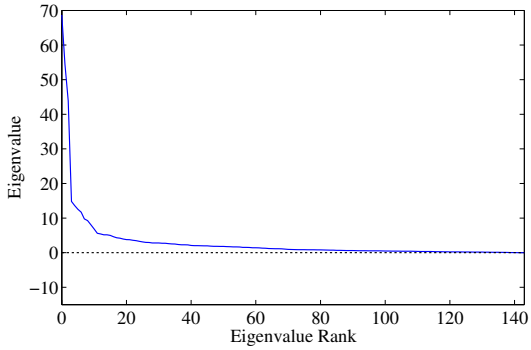
(b) Kernel matrix K_α



(c) Spectrum of S

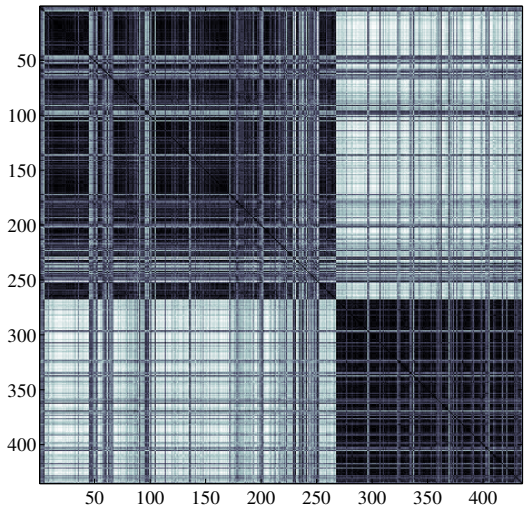


(d) Spectrum modification vector α

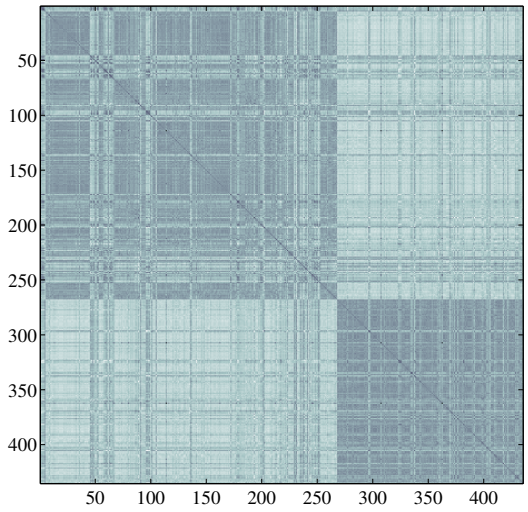


(e) Spectrum of K_α

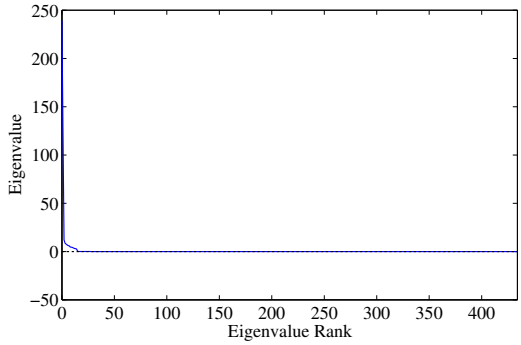
Figure 5.4: The similarity matrix S of the Protein-2 data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.



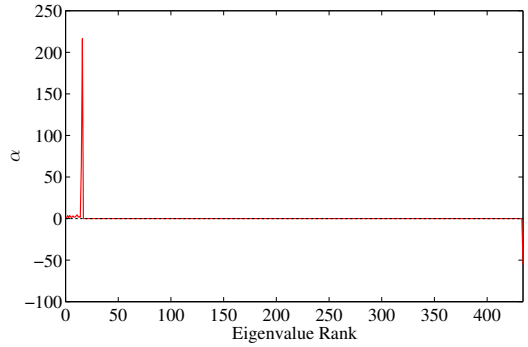
(a) Similarity matrix S



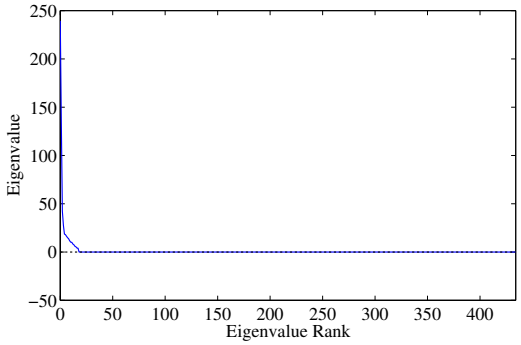
(b) Kernel matrix K_α



(c) Spectrum of S

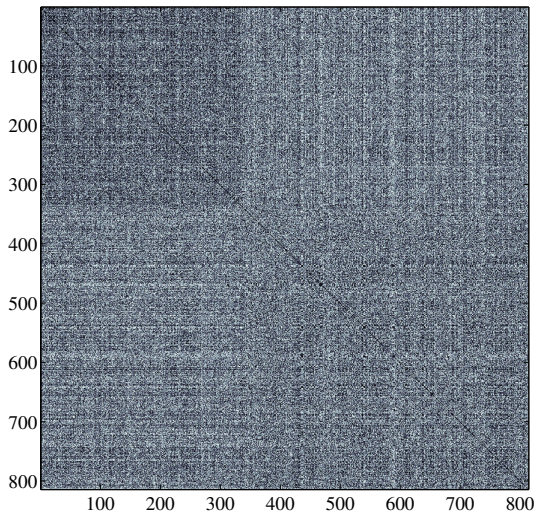


(d) Spectrum modification vector α

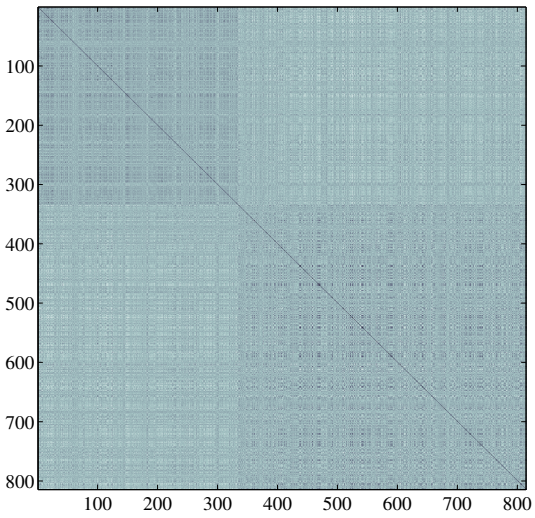


(e) Spectrum of K_α

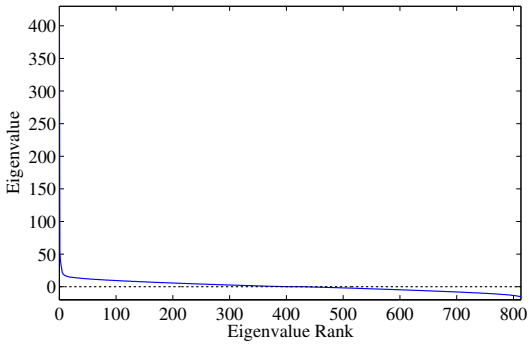
Figure 5.5: The similarity matrix S of the Voting data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.



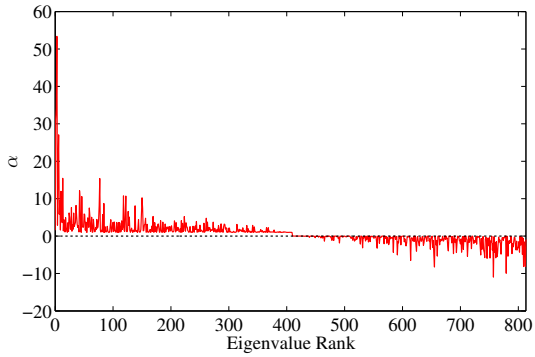
(a) Similarity matrix S



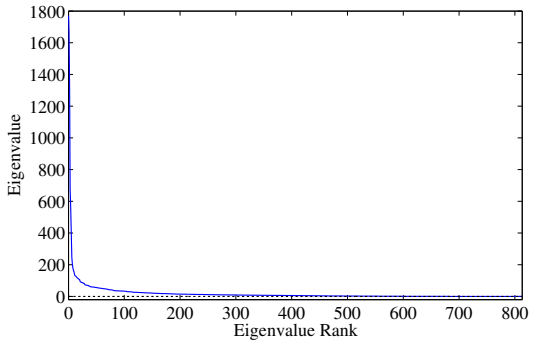
(b) Kernel matrix K_α



(c) Spectrum of S

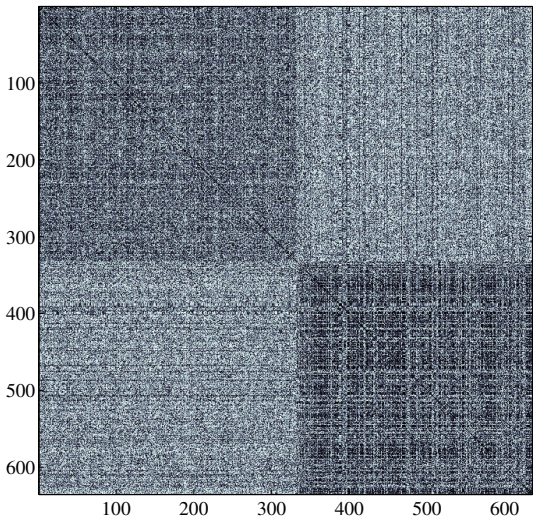


(d) Spectrum modification vector α

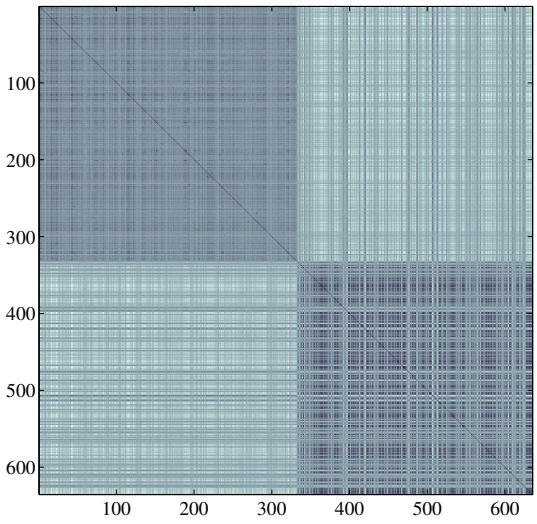


(e) Spectrum of K_α

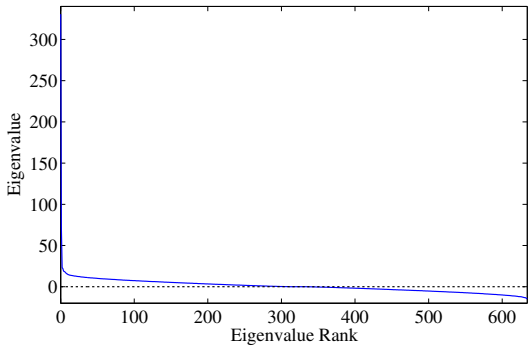
Figure 5.6: The similarity matrix S of the Yeast-5-7 data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.



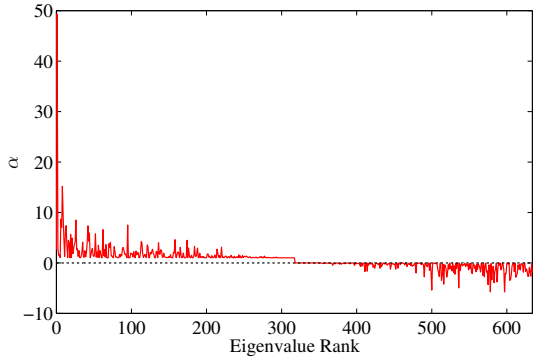
(a) Similarity matrix S



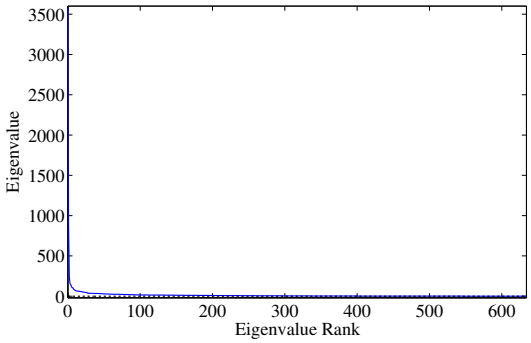
(b) Kernel matrix K_α



(c) Spectrum of S

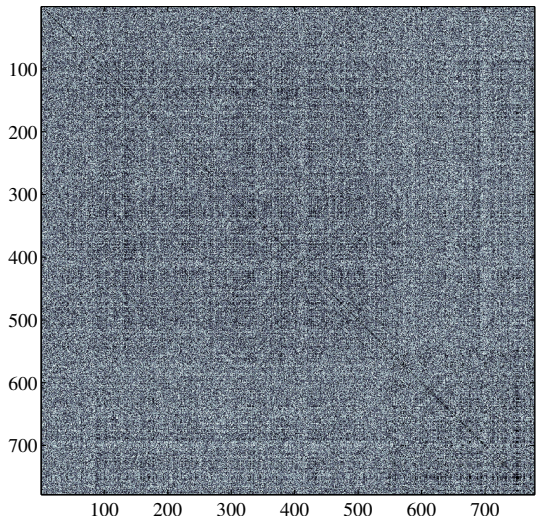


(d) Spectrum modification vector α

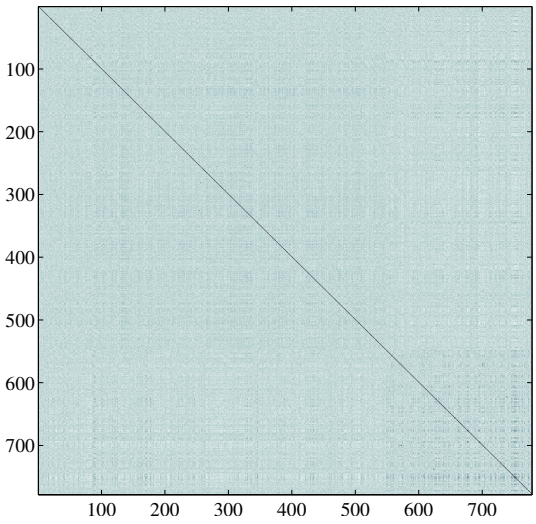


(e) Spectrum of K_α

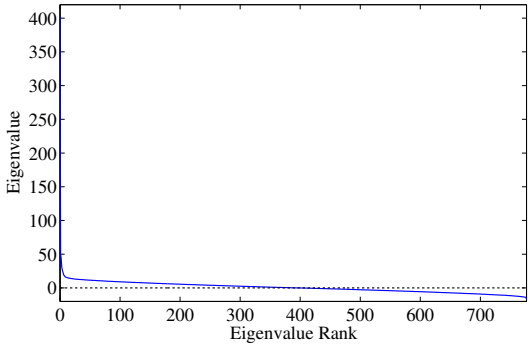
Figure 5.7: The similarity matrix S of the Yeast-5-12 data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.



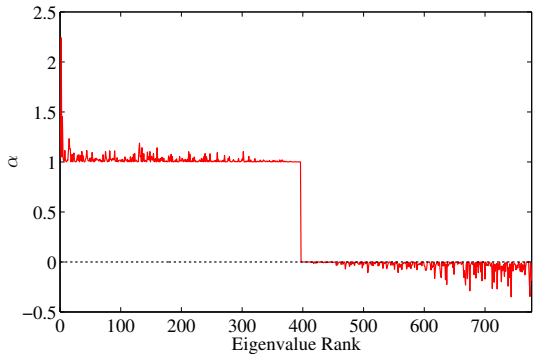
(a) Similarity matrix S



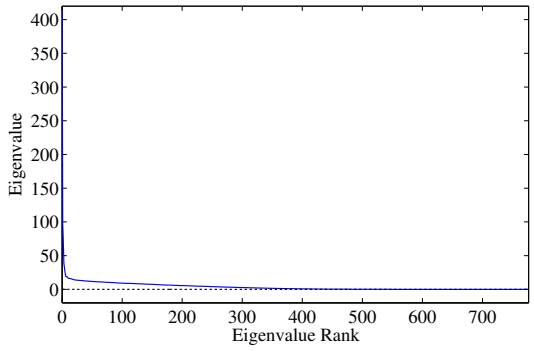
(b) Kernel matrix K_α



(c) Spectrum of S

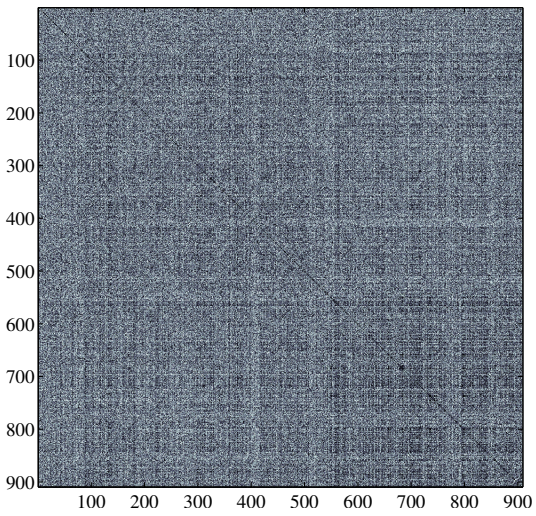


(d) Spectrum modification vector α

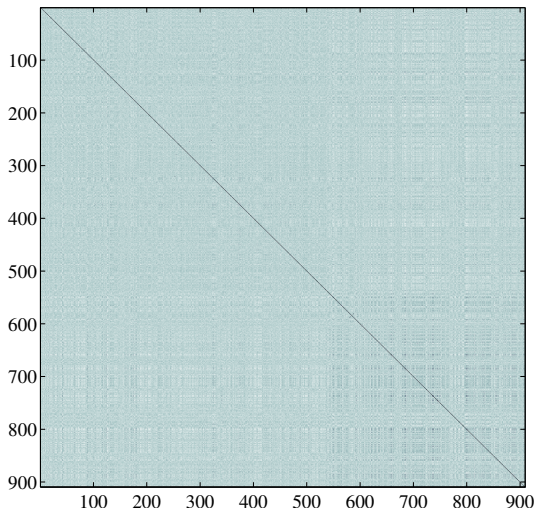


(e) Spectrum of K_α

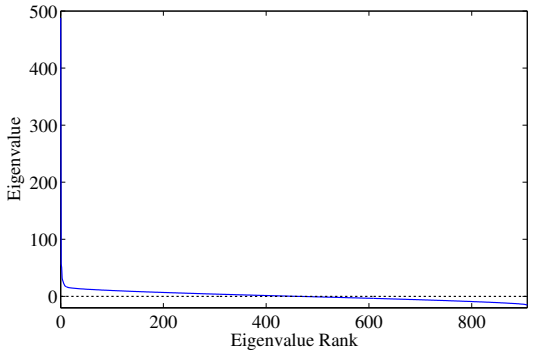
Figure 5.8: The similarity matrix S of the Yeast-6-8 data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.



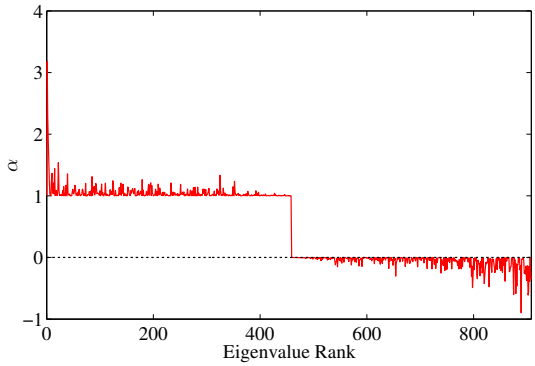
(a) Similarity matrix S



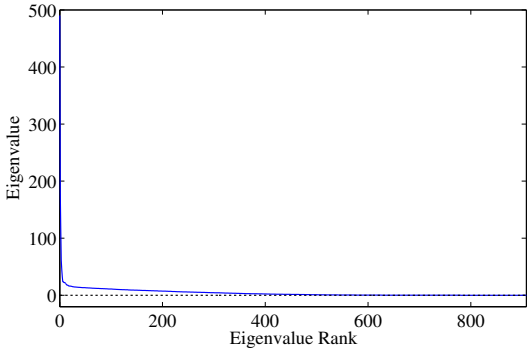
(b) Kernel matrix K_α



(c) Spectrum of S



(d) Spectrum modification vector α



(e) Spectrum of K_α

Figure 5.9: The similarity matrix S of the Yeast-6-10 data set and the kernel matrix K_α learned by the SimSVM are shown in (a) and (b), respectively. Also shown are the spectra of S and K_α, and the spectrum modification vector α.

properties of a kernel. For this reason, after a brief review of MKL, we discuss an extension of the SimSVM that combines an indefinite similarity with multiple kernels to produce a classifier with good generalization. Including indefinite similarities in the framework of kernel fusion provides a more comprehensive picture of the relationships between data samples and extends the concept of MKL.

5.4.1 Multiple Kernel Learning

MKL enables kernel methods to learn for a particular learning problem how to combine kernels with different parameters (for example, Gaussian RBF kernel with different bandwidths) or kernels from different sources.

Given m kernel matrices $K_1, \ldots, K_m$ each of size $n \times n$, consider the set of PSD matrices that are linear combinations of these m kernel matrices and have a fixed trace τ, that is,

$$\mathcal{K} = \left\{ K \succeq 0 \;\middle|\; K = \sum_{i=1}^{m} w_i K_i,\ w_i \in \mathbb{R},\ i = 1, \ldots, m,\ \operatorname{tr}(K) = \tau \right\}.$$

Let $\vartheta(K)$ be the optimal value of the SVM dual problem given in (2.4) for a specific kernel K. Lanckriet et al. proposed in [55] to learn an optimal linear combination of $K_1, \ldots, K_m$ by solving

$$\begin{aligned} \underset{K}{\text{minimize}} \quad & \vartheta(K) \\ \text{subject to} \quad & K \in \mathcal{K}. \end{aligned} \tag{5.32}$$

A thorough and detailed theoretical analysis of the above problem can be found in [69]. Since the SVM primal problem given in (2.3) and its dual have the same optimal value, based on the interpretation of the SVM primal problem, we can see that conceptually, (5.32) is equivalent to:

$$\begin{aligned} \underset{f, K}{\text{minimize}} \quad & \frac{1}{n} \sum_{i=1}^{n} L_{\text{hinge}}(y_i, f(x_i)) + \eta \|f\|_K^2 \\ \text{subject to} \quad & f \in \mathcal{H}_K, \quad K \in \mathcal{K}. \end{aligned}$$

Lanckriet et al. showed that the problem described by (5.32) is convex in K and can be

formulated as the following SDP:

$$\begin{aligned}
\underset{w,t,\mu,\nu,\delta}{\text{minimize}} \quad & t \\
\text{subject to} \quad & \operatorname{tr}\left(\sum_{i=1}^{m} w_i K_i\right) = \tau, \\
& \sum_{i=1}^{m} w_i K_i \succeq 0, \\
& \begin{bmatrix} \operatorname{diag}(y)\left(\sum_{i=1}^{m} w_i K_i\right)\operatorname{diag}(y) & \mathbf{1} + \mu - \nu + \delta y \\ (\mathbf{1} + \mu - \nu + \delta y)^T & t - 2C\mathbf{1}^T\nu \end{bmatrix} \succeq 0, \\
& \mu \geq 0, \quad \nu \geq 0,
\end{aligned}$$

with variables $w \in \mathbb{R}^m$, $t \in \mathbb{R}$, $\mu \in \mathbb{R}^n$, $\nu \in \mathbb{R}^n$ and $\delta \in \mathbb{R}$. By restricting the linear combination $\sum_{i=1}^{m} w_i K_i$ to be a conic combination such that $w_i \geq 0$, $i = 1, \ldots, m$, they further simplified the problem to the following QCQP:

$$\begin{aligned}
\underset{\alpha,t}{\text{maximize}} \quad & \mathbf{1}^T\alpha - \frac{1}{2}\tau t \\
\text{subject to} \quad & \alpha^T \operatorname{diag}(y) K_i \operatorname{diag}(y)\alpha \leq \operatorname{tr}(K_i)t, \quad i = 1, \ldots, m, \\
& 0 \leq \alpha \leq C\mathbf{1}, \\
& y^T\alpha = 0,
\end{aligned} \tag{5.33}$$

with variables $\alpha \in \mathbb{R}^n$ and $t \in \mathbb{R}$. To make (5.33) look more like (2.4), one can rewrite (5.33) by moving the m quadratic inequality constraints into the objective so that the problem becomes

$$\begin{aligned}
\underset{\alpha,t}{\text{maximize}} \quad & \mathbf{1}^T\alpha - \frac{1}{2}\max_i \left(\frac{\tau}{\operatorname{tr}(K_i)}\alpha^T \operatorname{diag}(y) K_i \operatorname{diag}(y)\alpha\right) \\
\text{subject to} \quad & 0 \leq \alpha \leq C\mathbf{1}, \\
& y^T\alpha = 0.
\end{aligned} \tag{5.34}$$

Although the objective function in (5.34) is concave, it is not differentiable, and thus the SMO algorithm [77, 78] for solving (2.4) cannot be applied. In [6], Bach et al. derived an equivalent problem to (5.34); by adding additional regularization terms, they were able to make the objective function of the equivalent problem differentiable, and then they proposed an SMO-like algorithm to solve it.

For efficiently solving MKL, two fast algorithms designed for large-scale MKL problems were proposed in [92] and [79] for a slightly different case where $\mathcal{K}$ is replaced by the set of convex combinations of $K_1, \ldots, K_m$, that is,

$$\mathcal{K}' = \left\{ K = \sum_{i=1}^{m} w_i K_i \;\middle|\; \mathbf{1}^T w = 1,\ w \geq 0 \right\}.$$

An extension of MKL to multiclass classification can be found in [111]. Another extension of MKL, termed localized MKL, was proposed in [36], where the weights w_i, $i = 1, \ldots, m$, are modeled as functions of samples. The relationship between MKL and group lasso [107] was established in [5], and recently an algorithm to solve MKL based on group lasso was proposed in [105].

5.4.2 Multiple Kernel Learning with an Indefinite Similarity

We assume that besides the m given kernel matrices $K_1, \ldots, K_m$, we are also given an indefinite similarity matrix S. The m kernel matrices $K_1, \ldots, K_m$ already have associated RKHS's: $\mathcal{H}_1, \ldots, \mathcal{H}_m$. For the indefinite similarity matrix S, we would like to find a surrogate kernel matrix K_0 corresponding to an RKHS denoted by $\mathcal{H}_0$. Then we can define a new RKHS $\mathcal{H}$ as the direct sum of $\mathcal{H}_i$, $i = 0, \ldots, m$, that is,

$$\mathcal{H} = \bigoplus_{i=0}^{m} \mathcal{H}_i,$$

and its associated inner product for any $a, b \in \mathcal{H}$ is defined for some $w_i \geq 0$, $i = 0, \ldots, m$, as

$$\langle a, b \rangle_{\mathcal{H}} = \sum_{i=0}^{m} w_i \langle a_i, b_i \rangle_{\mathcal{H}_i},$$

where a_i and b_i are the unique components of a and b in $\mathcal{H}_i$, respectively. Our goal is to learn a classifier in $\mathcal{H}$ that can generalize better than one trained in any single $\mathcal{H}_i$, $i = 0, \ldots, m$. To achieve this goal, we need to find an effective $\mathcal{H}_0$ and equip the inner product of $\mathcal{H}$ with an optimal set of weights w_i, $i = 0, \ldots, m$.

Same as the SimSVM, we let the surrogate kernel matrix be a spectrum modification of the indefinite similarity matrix:

$$K_0 = U \operatorname{diag}(\alpha) U^T S = U \operatorname{diag}(\alpha) U^T U \Lambda U^T = U \operatorname{diag}(\alpha) \Lambda U^T.$$

Then we combine K_0 with a conic combination of the m given kernel matrices, and form

$$\kappa(\alpha, w) = K_0 + \sum_{i=1}^{m} w_i K_i = U \operatorname{diag}(\alpha) \Lambda U^T + \sum_{i=1}^{m} w_i K_i,$$

where $w_i \geq 0$, $i = 1, \ldots, m$. Let $G_i = u_i u_i^T$, $i = 1, \ldots, n$, where u_i is the ith column of U, and we can rewrite $\kappa(\alpha, w)$ as

$$\kappa(\alpha, w) = \sum_{i=1}^{n} \lambda_i \alpha_i G_i + \sum_{j=1}^{m} w_j K_j.$$

Due to the constraints $\lambda_i \alpha_i \geq 0$, $i = 1, \ldots, n$, the above expression implies that $\kappa(\alpha, w)$ is actually a conic combination of $(m + n)$ kernels: $|\lambda_1| G_1, \ldots, |\lambda_n| G_n$, $K_1, \ldots, K_m$, and their corresponding weights are $|\alpha_1|, \ldots, |\alpha_n|, w_1, \ldots, w_m$.

We propose to minimize the empirical risk with regularization simultaneously over the spectrum modification vector $\alpha \in \mathbb{R}^n$, the kernel conic combination weights $w \in \mathbb{R}^m$, and the SVM parameters by extending (5.20) to:

$$\begin{aligned} \underset{c, b, \xi, \alpha, w}{\text{minimize}} \quad & \frac{1}{n} \mathbf{1}^T \xi + \eta c^T \kappa(\alpha, w) c + \gamma h(\alpha) \\ \text{subject to} \quad & \operatorname{diag}(y) \left(\kappa(\alpha, w) c + b \mathbf{1} \right) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \quad w \geq 0, \\ & \Lambda \alpha \geq 0, \\ & \sum_{i=1}^{m} w_i \operatorname{tr}(K_i) \leq \tau, \end{aligned} \tag{5.35}$$

where $\tau > 0$ is a constant. We refer to (5.35) as the *SimMKL*. The trace constraint on the last line of (5.35) is a linear inequality constraint on w, which prevents w from growing unbounded. This constraint also guarantees that the inner product in $\mathcal{H}$ is bounded, which is a crucial condition in Theorem 2.1 for establishing a generalization bound.

We can use the same technique that we used in Section 5.1 to reformulate (5.35) as a convex optimization problem. First, we let $z = \kappa(\alpha, w) c$. Then by introducing slack

variables $u, v \in \mathbb{R}$ and applying Lemma 5.1, we can rewrite (5.35) as

$$\begin{aligned}
\underset{z,b,\xi,\alpha,w,u,v}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta u + \gamma v \\
\text{subject to} \quad & \operatorname{diag}(y)(z + b\mathbf{1}) \geq \mathbf{1} - \xi, \\
& \xi \geq 0, \quad w \geq 0, \\
& \Lambda\alpha \geq 0, \\
& \sum_{i=1}^{m} w_i \operatorname{tr}(K_i) \leq \tau, \\
& \begin{bmatrix} \sum_{i=1}^{n} \lambda_i \alpha_i G_i + \sum_{j=1}^{m} w_j K_j & z \\ z^T & u \end{bmatrix} \succeq 0, \\
& h(\alpha) \leq v.
\end{aligned} \tag{5.36}$$

By solving the convex optimization problem given by (5.36), we jointly (i) learn the spectrum modification on the similarity matrix to make it PSD, (ii) learn the optimal conic combination of the m given kernel matrices, and (iii) learn the parameters of an SVM that acts on this conic combination of PSD matrices.

Let $z^\star$, $b^\star$, $\alpha^\star$ and $w^\star$ denote the optimal solution to (5.36). We can recover the optimal $c^\star$ by

$$c^\star = \left(\kappa(\alpha^\star, w^\star)\right)^\dagger z^\star.$$

For a test samples x, given its similarity vector $s \in \mathbb{R}^n$ and m kernel vectors:

$$k_i = \begin{bmatrix} K_i(x, x_1) & \dots & K_i(x, x_n) \end{bmatrix}^T, \quad i = 1, \dots, m,$$

we classify x as

$$\hat{y} = \operatorname{sgn}\left((c^\star)^T \left(U \operatorname{diag}(\alpha^\star) U^T s + \sum_{i=1}^{m} w_i^\star k_i \right) + b^\star \right).$$

5.4.3 Experiments

We compare the proposed SimMKL for data fusion with SVMs using a surrogate kernel for the indefinite similarity formed by spectrum clip, flip or shift, and with SVMs trained individually on one of the given kernels.

Four real data sets were used for experiments. We ran one experiment each with the Amazon-2, Aural Sonar, and Protein-2 data sets, and two experiments with the Yeast-7-12 data set. These data sets have already been described in Section 5.3.2 except the Yeast-7-12 data set, which was formed by selecting from the original Yeast data set [56] the first 100 samples that exclusively belong to class 7 and the first 100 samples that exclusively belong to class 12.

For the Amazon-2 data set, we created three kernels from its similarities. The first two were created by treating the similarities as features. Let s_j denote the jth column of the similarity matrix S of all the samples in that data set. The first kernel is a linear kernel on similarity features:

$$K_1(x_i, x_j) = s_i^T s_j,$$

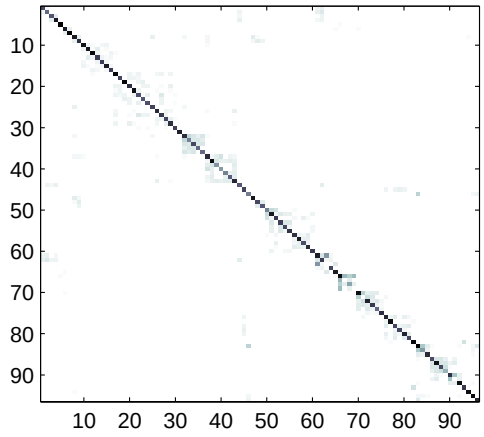
and the second kernel is a Gaussian RBF kernel on similarity features:

$$K_2(x_i, x_j) = \exp\left(-\beta \|s_i - s_j\|_2^2\right),$$

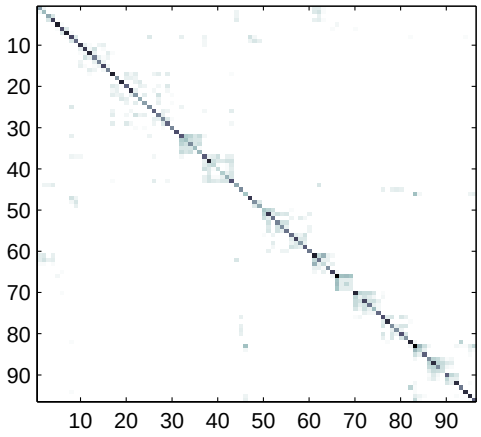
with $\beta = 0.1$. Given that the original similarity matrix of this data set is very sparse, we created the third kernel by treating S as the adjacency matrix of a weighted graph and generated a diffusion kernel [53, 91] using the normalized graph Laplacian with parameter $\sigma^2 = 20$. These kernel matrices are shown in Figure 5.10.

For the Aural Sonar data set, we created all three kernels by treating the similarities as features. The first one is a linear kernel, and the other two are Gaussian RBF kernels with $\beta = 0.01$ and $\beta = 0.1$, respectively. We repeated the same process for the Protein-2 data set except that for the two Gaussian RBF kernels, we chose $\beta = 0.1$ and $\beta = 0.05$. The kernel matrices of these two data sets can be found in Figure 5.11 and 5.12, respectively.

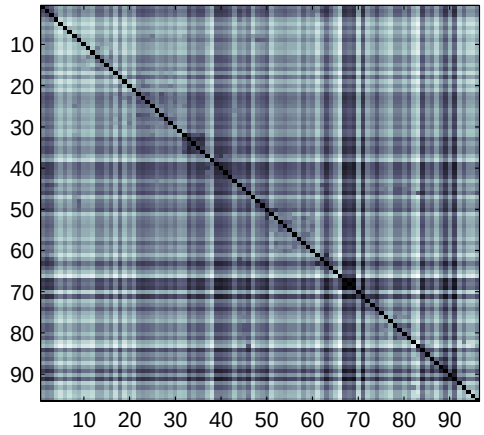
For the Yeast-7-12 data set, the indefinite similarity here is the Smith-Waterman E-value. We created three kernels by treating the similarities as features: one is a linear kernel and the other two are Gaussian RBF kernels with $\beta = 0.01$ and $\beta = 0.001$, respectively. We added a fourth kernel by using the Pfam kernel in [56], which measures the similarities between the yeast proteins in a different way than the Smith-Waterman algorithm. These kernel matrices are shown in Figure 5.13. We ran two sets of experiments on this data set.



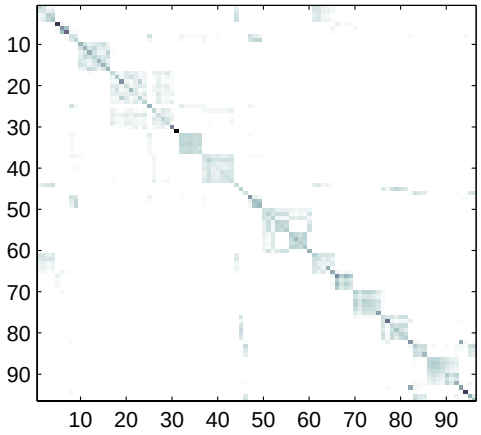
(a) Similarity matrix S



(b) Kernel matrix K_1

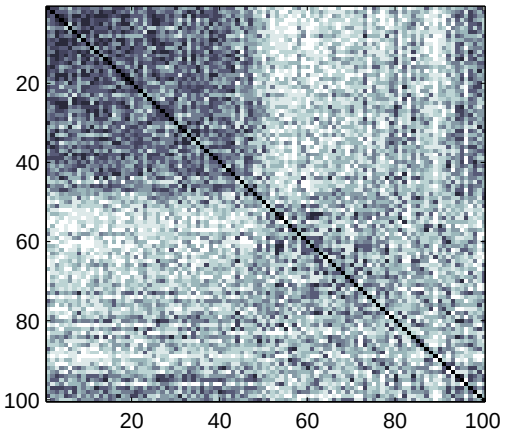


(c) Kernel matrix K_2

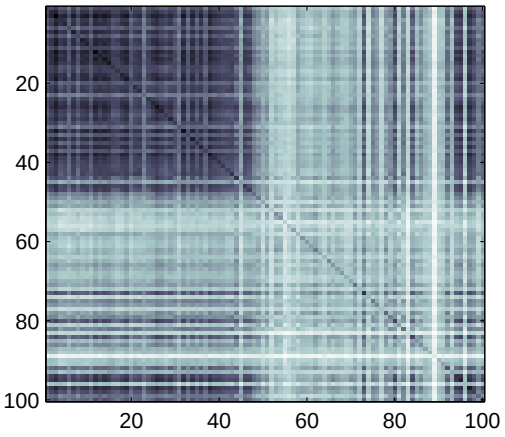


(d) Kernel matrix K_3

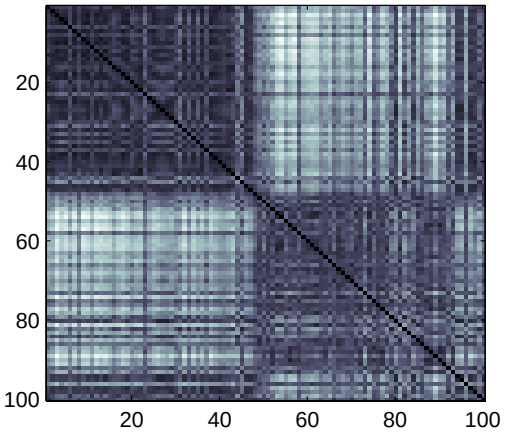
Figure 5.10: Similarity and kernel matrices of the Amazon-2 data set. K_1 is a linear kernel on similarity features. K_2 is a Gaussian RBF kernel on similarity features. K_3 is a diffusion kernel computed on a weighted graph using S as its adjacency matrix.



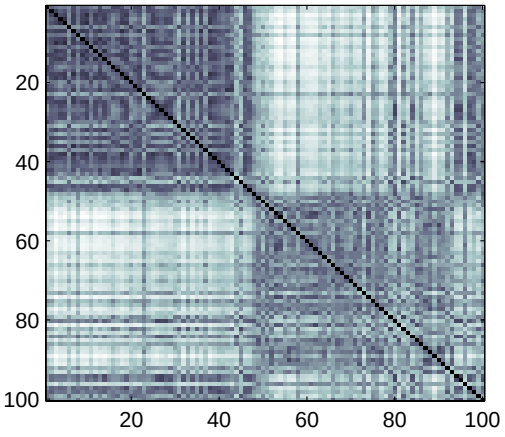
(a) Similarity matrix S



(b) Kernel matrix K_1

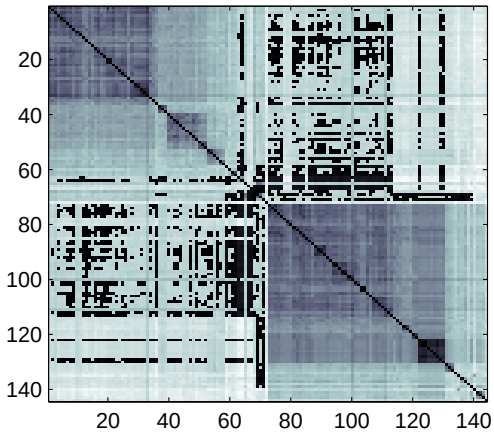


(c) Kernel matrix K_2

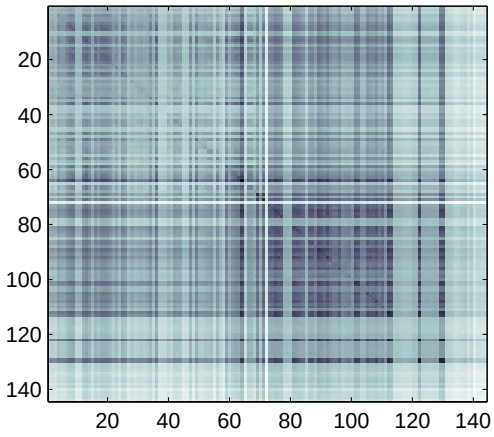


(d) Kernel matrix K_3

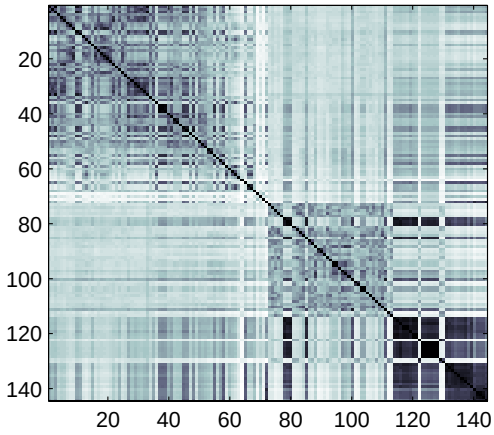
Figure 5.11: Similarity and kernel matrices of the Aural Sonar data set. K_1 is a linear kernel on similarity features. K_2 and K_3 are two Gaussian RBF kernels on similarity features with different bandwidths.



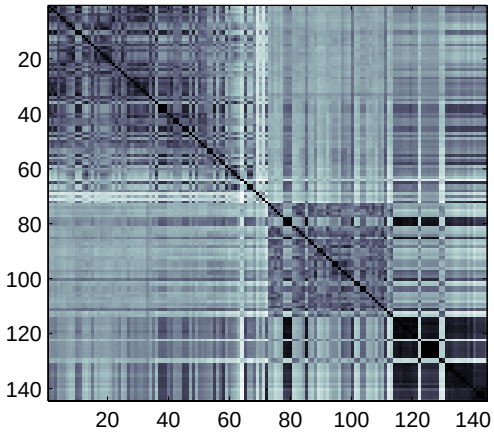
(a) Similarity matrix S



(b) Kernel matrix K_1

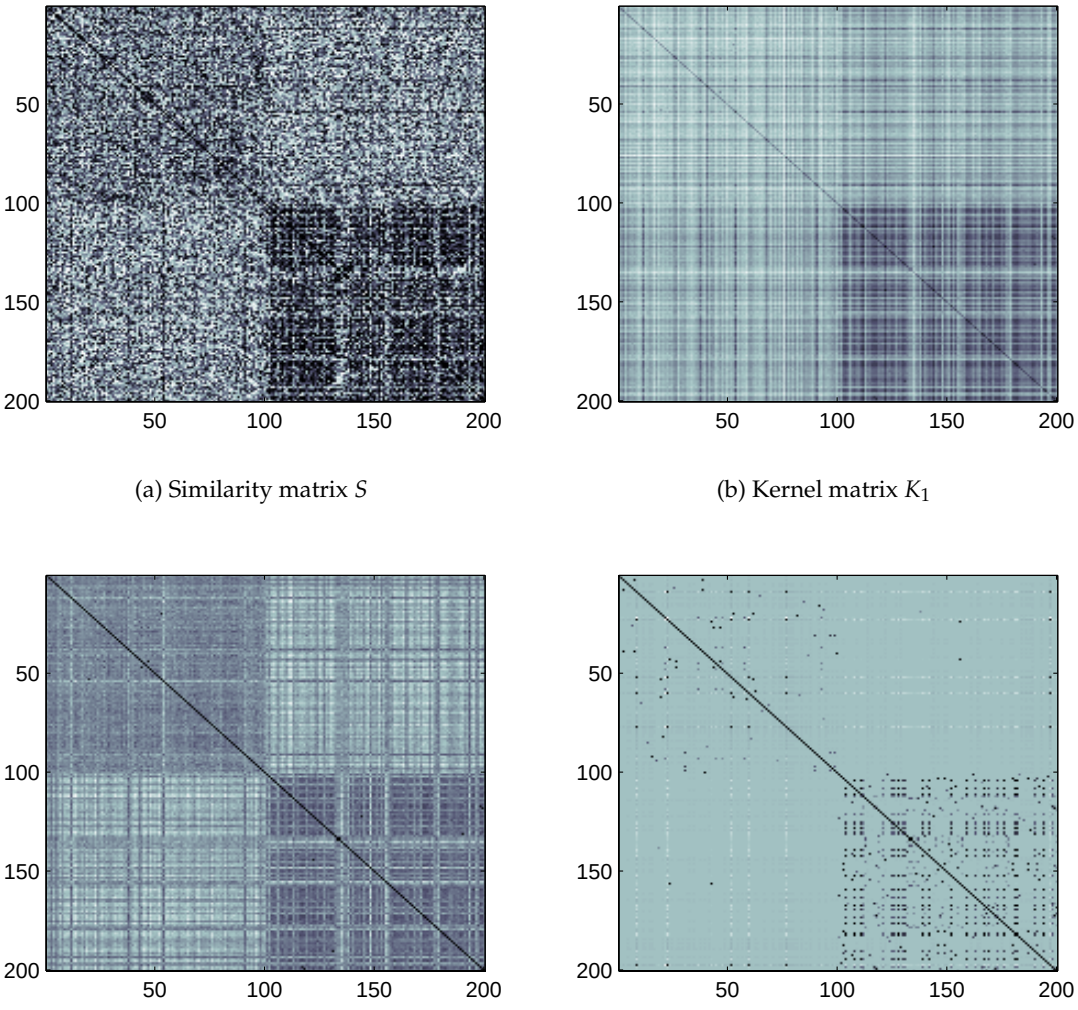


(c) Kernel matrix K_2



(d) Kernel matrix K_3

Figure 5.12: Similarity and kernel matrices of the Protein-2 data set. K_1 is a linear kernel on similarity features. K_2 and K_3 are two Gaussian RBF kernels on similarity features with different bandwidths.



(a) Similarity matrix S

(b) Kernel matrix K_1

(c) Kernel matrix K_3

(d) Kernel matrix K_{Pfam}

Figure 5.13: Similarity and kernel matrices of the Yeast-7-12 data set. K_1 is a linear kernel on similarity features. K_3 is a Gaussian RBF kernel on similarity features. K_{Pfam} is the Pfam kernel. Because K_2 looks highly similar to K_3, it is not shown here.

Table 5.5: Mean and standard deviation (in parentheses) of the test errors (in percentage) across 20 randomized test/training partitions for the Amazon-2, Aural Sonar and Protein-2 data sets. For each data set, the lowest mean error and those not statistically significantly worse are boldfaced.

	Amazon-2		Aural Sonar		Protein-2	
SimMKL	**9.74**	(7.30)	**12.00**	(6.16)	**1.38**	(2.35)
SVM w/ clip	11.84	(7.80)	**13.00**	(5.48)	8.79	(5.06)
SVM w/ flip	20.00	(11.66)	13.25	(5.45)	4.83	(3.04)
SVM w/ shift	17.89	(12.00)	32.75	(18.95)	26.55	(7.68)
SVM w/ kernel 1	11.05	(11.05)	**13.50**	(7.45)	3.45	(3.16)
SVM w/ kernel 2	12.89	(8.27)	13.75	(7.23)	**1.55**	(2.37)
SVM w/ kernel 3	**9.21**	(7.61)	13.75	(6.66)	**1.90**	(2.85)

One was to fuse the Smith-Waterman similarity and the three kernels created out of it, and the other was to fuse the Smith-Waterman similarity and the Pfam kernel.

The experimental setup for the classification test was exactly the same as what we have described in Section 5.3.2. For the SimMKL, we chose the regularizer $h(\alpha) = \left\|\alpha - \alpha_{\text{clip}}\right\|_2$ for all the data sets. With this regularizer, (5.36) becomes a convex conic program, and we solved it by the semidefinite-quadratic-linear program solver SDPT3 [97]. The hyperparameter C for the C-SVM and the regularization parameters η and γ for the SimMKL were cross-validated from the following sets:

$$
\begin{aligned}
C &\in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}, \\
\eta &\in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}, \\
\gamma &\in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}.
\end{aligned}
$$

The test errors averaged over 20 randomized test/training partitions are shown in Table 5.5 and 5.6. For each classification task, the lowest average error is boldfaced. Also boldfaced are the results that are not statistically significantly worse than the lowest one

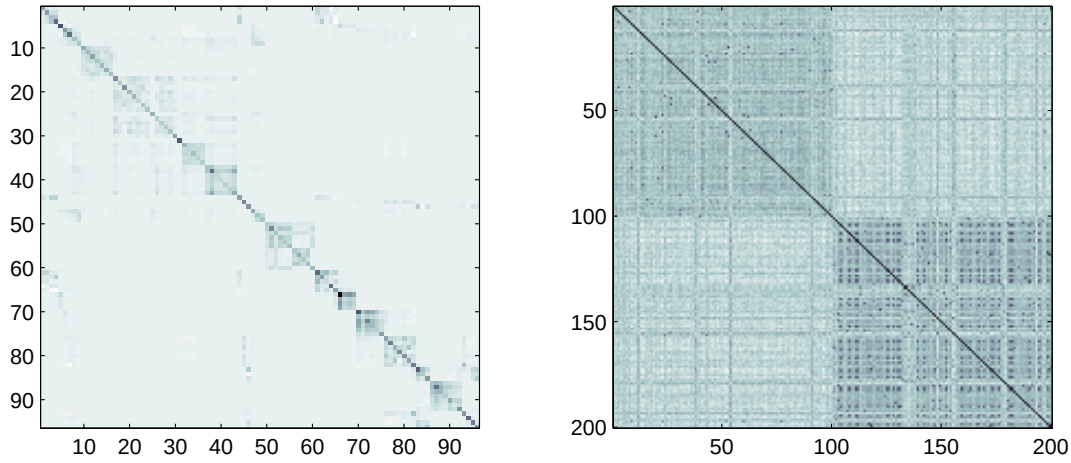
Table 5.6: Mean and standard deviation (in parentheses) of the test errors (in percentage) across 20 randomized test/training partitions for the two tasks on the Yeast-7-12 data set. The first task used the Smith-Waterman (SW) similarity and three kernels created out of it, and the second task used the Smith-Waterman similarity and the Pfam kernel. For each task, the lowest mean error and those not statistically significantly worse are boldfaced.

	Yeast-7-12 SW		Yeast-7-12 SW-Pfam	
SimMKL	**7.38**	(3.67)	**6.13**	(4.25)
SVM w/ clip	**7.25**	(3.53)	**7.25**	(3.53)
SVM w/ flip	**8.00**	(3.86)	8.00	(3.86)
SVM w/ shift	43.50	(8.37)	43.50	(8.37)
SVM w/ kernel 1	**8.38**	(3.06)	14.88	(5.16)
SVM w/ kernel 2	**8.00**	(3.40)	–	–
SVM w/ kernel 3	**7.63**	(4.09)	–	–

based on a one-sided Wilcoxon signed-rank test at the 5% significance level.[4] The results show that the proposed SimMKL achieves the lowest average error in three out of the five experiments, and unlike any of the other methods, in all the experiments the SimMKL is either the best or not statistically significantly different from the best. Interestingly, one can see from the last column of Table 5.6 that the average error of the SVM using the Pfam kernel alone is twice as high as that of the SVM using spectrum clip on the Smith-Waterman similarity, yet fusing these two descriptions can in fact improve the classification performance. This clearly demonstrates the potential benefits of combining multiple descriptions from different sources.

Moreover, we illustrate the fused similarity $\kappa(\alpha^\star, w^\star)$ learned by the SimMKL for the Amazon-2 and Yeast-7-12 (Smith-Waterman and Pfam fusion) data sets in Figure 5.14, where for each data set we trained the SimMKL on all the samples using the parameters

[4] A statistical significance test decides whether a classifier performs consistently better or worse than another classifier, and the result may differ from that indicated by the average performance. For example, the results on the Aural Sonar data set show that the SVM with spectrum flip is statistically significantly worse than the SimMKL but the SVM trained on kernel 1 is not, although the average error of the former is less than that of the latter.



(a) Amazon-2

(b) Yeast-7-12 SW-Pfam

Figure 5.14: Fused kernel matrices learned from the Amazon-2 and Yeast-7-12 data sets.

selected most frequently over the 20 random partitions. For the Amazon-2 data set, the intraclass similarities have been enhanced, as can be seen from the more obvious block-diagonal structure. For the Smith-Waterman and Pfam fusion, the fused kernel matrix appears to be a less noisy version of the original indefinite similarity matrix shown in Figure 5.13(a) with some scattered contributions from the Pfam kernel.

Chapter 6

KERNEL MATRIX COMPLETION USING AUXILIARY INFORMATION

An educated person is one who has learned that information almost always turns out to be at best incomplete and very often false, misleading, fictitious, mendacious — just dead wrong.

Russell Baker

So far we have assumed that all pairwise similarities are available. This may not be the case in some applications, where some similarities are simply missing. These missing similarities lead to missing entries in the similarity matrix S. In general, there are two typical cases of the locations of the missing entries. One is that the locations of the missing entries are distributed uniformly, that is, each (i, j)-entry has equal probability being missing. The other is that the similarity matrix S has entire rows or columns missing. Suppose a similarity is computed from some measurements of the samples. If such measurements are not available for some samples, then no similarity involving any of those samples can be computed, resulting in missing rows and missing columns in S.

For the first case, that is, uniformly distributed missing entries, Graepel specifically considered kernels and proposed to complete the kernel matrix K by solving the following problem [37]:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \sum_{(i,j)\in\mathcal{I}} (X_{ij} - K_{ij})^2 \\ \text{subject to} \quad & X \succeq 0, \end{aligned} \tag{6.1}$$

where $\mathcal{I}$ denotes the set of the locations of all known entries in K. The above problem can be easily formulated as an SDP. One can extend (6.1) to finding a complete kernel matrix from any incomplete similarities by replacing the K_{ij}'s with the known S_{ij}'s. Note that the solution to (6.1) might not be unique. If the complete similarity matrix is low-rank or approximately low-rank, according to the recent results of the research on matrix

completion [15], one should be able to recover the complete similarity matrix S by solving

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \|X\|_* \\ \text{subject to} \quad & X_{ij} = S_{ij}, \quad \forall\,(i,j) \in \mathcal{I}, \end{aligned} \tag{6.2}$$

where $\|X\|_*$ is the nuclear norm of X, which is defined as the sum of all the singular values of X, that is, $\|X\|_* = \sum_{i=1}^{n} \sigma_i(X)$. The problem in (6.2) can be rewritten as an SDP [27], and for completing a kernel matrix, the SDP is simply

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \operatorname{tr}(X) \\ \text{subject to} \quad & X_{ij} = K_{ij}, \quad \forall\,(i,j) \in \mathcal{I}, \\ & X \succeq 0. \end{aligned}$$

If the known entries in the similarity matrix S are noisy observations, then instead of (6.2), one can solve the following problem [14]:

$$\underset{X}{\text{minimize}} \quad \sum_{(i,j)\in\mathcal{I}} (X_{ij} - S_{ij})^2 + \mu\|X\|_*,$$

where $\mu > 0$ is a regularization parameter.

This chapter focuses on the second case, and specifically on incomplete kernel matrices with missing rows and columns. Without loss of generality, we assume that for an $n \times n$ kernel matrix

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix},$$

only the upper left block $K_{11} \in \mathbb{R}^{n_1 \times n_1}$ is known, where $n_1 < n$, and the other three blocks are missing. Without any additional information, it is impossible to infer the missing entries from K_{11} since there exist an endless number of possibilities for K_{12}, K_{21} and K_{22} to make K PSD. Therefore, we assume that we are also given a complete auxiliary kernel matrix $K_{\text{aux}} \in \mathbb{R}^{n \times n}$ for the same set of samples, and K_{aux} is derived from some measurements of the samples that are easier to obtain than those used to compute K. For example, for proteins, K can be a kernel matrix derived from measurements of the three-dimensional structure (also known as the tertiary structure) of proteins, while K_{aux} can be a kernel matrix derived from proteins' sequence information; to measure the 3D coordinates of the

atoms in a protein is known to be more difficult and expensive than to sequence a protein's amino acid composition [50]. Another example is that the measurements for K require expensive, high-resolution sensors, while the measurements for K_{aux} are obtained by noisy, low-resolution sensors. The matrix completion problem discussed here is to complete the kernel matrix K given K_{11} and K_{aux} so that we can do better in our learning tasks using K than using either K_{11} or K_{aux}.

We first review the prior work in Section 6.1, and then in Section 6.2 we discuss in detail the problem settings for both inductive and transductive learning. Section 6.3 sketches one possible direction for future research.

6.1 Prior Work

To my best knowledge, the *em* algorithm[1] [96] is the first method proposed for completing K using an auxiliary kernel matrix K_{aux}. The authors of [96] assumed $K_{11} \succ 0$, that is, K_{11} is strictly positive definite, and they formed two convex sets:

$$\mathcal{K} = \left\{ K \in \mathbb{R}^{n \times n} \;\middle|\; K \succ 0,\ K_{ij} = (K_{11})_{ij},\ i, j = 1, \ldots, n_1 \right\},$$

and

$$\mathcal{M} = \left\{ M \in \mathbb{R}^{n \times n} \;\middle|\; M = \sum_{i=1}^{n} w_i M_i,\ w_i > 0,\ i = 1, \ldots, n \right\}, \tag{6.3}$$

where $M_i = v_i v_i^T$ and v_i is the ith eigenvector of K_{aux} for a certain eigenvalue decomposition. The *em* algorithm is an iterative procedure to solve

$$\begin{aligned} \underset{K,\, M}{\text{minimize}} \quad & L_{\text{S}}(K, M) \\ \text{subject to} \quad & K \in \mathcal{K}, \quad M \in \mathcal{M}, \end{aligned} \tag{6.4}$$

where L_{S} is Stein's loss:[2]

$$L_{\text{S}}(K, M) = \operatorname{tr}(KM^{-1}) - \log \left| KM^{-1} \right| - n.$$

[1] The authors of [96] used *em* to emphasize that their proposed algorithm is different from the expectation-maximization (EM) algorithm for estimating parameters in statistical models.

[2] Note that Stein's loss is a Bregman divergence between positive definite matrices. Stein's loss is convex in K (because $f(X) = -\log|X|$ is convex for $X \succ 0$), but neither convex nor concave in M, though it is convex in M^{-1}.

At the ith iteration, the e-step solves

$$K^{(i)} = \arg\min_{K \in \mathcal{K}} L_S(K, M^{(i-1)}),$$

where $M^{(i-1)}$ is the solution from the previous iteration, and the following m-step solves

$$M^{(i)} = \arg\min_{M \in \mathcal{M}} L_S(K^{(i)}, M).$$

Both steps have closed-form solutions [96]. However, whether this alternating minimization procedure will converge to a point in the solution set of (6.4) remains an open question.[3] Suppose the *em* algorithm converges at the jth iteration, then $K^{(j)}$ will be used as the complete kernel matrix K. The construction of the convex set $\mathcal{M}$ given by (6.3) implies the assumption that the useful information provided by K_{aux} for recovering K is mainly carried by its eigenvectors. Although no theoretical or intuitive justification for this assumption was provided in [96], the experiments on bacteria clustering [96] and protein classification [50] showed that the kernel matrix K recovered by the *em* algorithm could indeed be more helpful to the learning task than K_{aux}.

In [49], Kato et al. extended the *em* algorithm to inferring networks of proteins. An example is to infer the metabolic network of yeast proteins, represented by an undirected graph. They assumed that part of the network, that is, a subgraph, was known and multiple types of auxiliary information, such as gene expression, phylogenetic profiles and amino acid sequences, were given for all the samples. They formulated the network inference problem as a kernel matrix completion problem by assuming that the edges of a graph can be (approximately) recovered by thresholding a diffusion kernel [53, 91] computed on that graph, that is, for any $i \neq j$,

$$A_{ij} = \begin{cases} 1, & K_{ij} \geq \delta, \\ 0, & K_{ij} < \delta, \end{cases}$$

where A is the adjacency matrix, K is the diffusion kernel matrix, and $\delta > 0$ is the threshold. They computed a diffusion kernel on the known subgraph and used that diffusion kernel

[3] The convergence of such alternating minimization method is proved in [10] for Bregman divergences that are jointly convex; nevertheless, Stein's loss function $L_S(K, M)$ is not jointly convex in K and M.

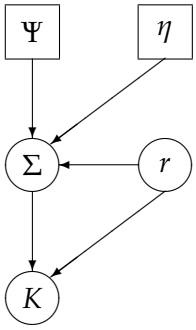


Figure 6.1: Hierarchical model proposed in [109] for kernel matrix completion. Each circle represents a random variable (or random matrix), and each square represents a fixed parameter (or parameter matrix).

matrix as K_{11}, even though this in general is not true. For the given m types of auxiliary information, they represented each of them by a kernel matrix $K_{\text{aux}}^{(i)}$, $i = 1, \ldots, m$. They modified the *em* algorithm to complete the diffusion kernel matrix K given K_{11} and $K_{\text{aux}}^{(i)}$, $i = 1, \ldots, m$, and then thresholded K to infer the edges that involve the nodes outside the known subgraph. In the modified *em* algorithm, instead of $\mathcal{M}$ given by (6.3), they formed the following convex set:

$$\mathcal{M}' = \left\{ M \in \mathbb{R}^{n \times n} \;\middle|\; M = \sum_{i=1}^{m} w_i K_{\text{aux}}^{(i)} + \sigma^2 I,\ \sum_{i=1}^{m} w_i = 1,\ w_i \geq 0,\ i = 1, \ldots, m \right\},$$

where $\sigma^2 > 0$ is a regularization parameter ensuring $M \succ 0$ for all $M \in \mathcal{M}'$.

Zhang et al. took a generative model approach for kernel matrix completion [109]. They proposed a hierarchical model shown in Figure 6.1. They first assumed that the kernel matrix K is drawn from a Wishart distribution,[4] that is, $K \sim \mathcal{W}_n(r, \Sigma/r)$. They further assumed that given r, the parameter matrix Σ is distributed according to an inverse Wishart distribution, that is,

$$\Sigma \,|\, r \sim \mathcal{IW}_n(\eta r + n + 1, \eta r \Psi),$$

where $\eta \in \mathbb{R}$ and $\Psi \in \mathbb{R}^{n \times n}$ are fixed parameters, and they let $\Psi = K_{\text{aux}}$. To infer the two

[4]The definitions of the Wishart and inverse Wishart distributions and some of their basic properties can be found in Appendix C. We refer the reader to [41] for more details on matrix variate distributions.

missing blocks K_{12} and K_{22} (notice that $K_{21} = K_{12}^T$), they first computed the maximum *a posteriori* (MAP) estimates of Σ and r:

$$\begin{aligned}(\hat{\Sigma}, \hat{r}) &= \arg\max_{\Sigma, r} p(\Sigma, r \mid K_{11}, K_{\text{aux}}, \eta) \\ &= \arg\max_{\Sigma, r} p(K_{11} \mid \Sigma, r, K_{\text{aux}}, \eta) p(\Sigma, r \mid K_{\text{aux}}, \eta) \\ &= \arg\max_{\Sigma, r} p(K_{11} \mid \Sigma, r) p(\Sigma \mid r, K_{\text{aux}}, \eta) p(r) \\ &= \arg\max_{\Sigma, r} p(K_{11} \mid \Sigma, r) p(\Sigma \mid r, K_{\text{aux}}, \eta),\end{aligned}$$

where they assumed a noninformative uniform prior on r. They solved the above problem using the EM algorithm. With the MAP estimates $\hat{\Sigma}$ and $\hat{r}$, they then recovered K_{12} and K_{22} by computing the conditional expectation of these two blocks given K_{11} with respect to the model $K \sim \mathcal{W}_n(\hat{r}, \hat{\Sigma}/\hat{r})$.

In [109], Zhang et al. also showed that if one modifies the generative model shown in Figure 6.1 by constraining $\Sigma \in \mathcal{M}$, then the EM algorithm for solving the following maximum likelihood estimation (MLE) problem:

$$\begin{aligned}\underset{\Sigma, r}{\text{maximize}} \quad & p(K_{11} \mid \Sigma, r) \\ \text{subject to} \quad & \Sigma \in \mathcal{M}, \\ & r > n - 1,\end{aligned}$$

is exactly the *em* algorithm. This result justifies the *em* algorithm from a generative model perspective.

6.2 Problem Settings

The classification problems considered in [50] and [109] were all under the transductive learning setting. For transductive learning, we have n samples in total, part of which is the training set, whose class labels are given. We denote the set of the indices of the training samples by $\mathcal{T}$. For the kernel matrix K, we are only given the upper left block K_{11} of size $n_1 \times n_1$, where $n_1 < n$. We are also given a complete auxiliary kernel matrix K_{aux} of size $n \times n$, which is supposed to be less informative in regard to the classification task than K, if

complete. The classification problem here is to estimate the class labels of the test samples, that is, those whose indices are in the set $\{1,\ldots,n\}\backslash\mathcal{T}$.

For inductive learning, we first have n training samples with known class labels represented by $y \in \mathbb{R}^n$. For these n training samples, we are given $K_{11} \in \mathbb{R}^{n_1\times n_1}$, which is the upper left block of the kernel matrix $K \in \mathbb{R}^{n\times n}$, and also an auxiliary kernel matrix $K_{\text{aux}} \in \mathbb{R}^{n\times n}$. We train a classifier using y, K_{11} and K_{aux}. The classification problem is to estimate the class labels of the test samples that will come later, and we assume that the test samples come one at a time. For a test sample x, we consider two cases based on the information we have about x. In the first case, we only have for x the measurements used to compute the auxiliary kernel function, and thus we only have $K_{\text{aux}}(x,x)$ and $K_{\text{aux}}(x,x_i)$, $i = 1,\ldots,n$. In the second case, besides $K_{\text{aux}}(x,x)$ and $K_{\text{aux}}(x,x_i)$, $i = 1,\ldots,n$, we also have the measurements used to compute K, and hence we also have $K(x,x)$ and $K(x,x_i)$, $i = 1,\ldots,n_1$ with $n_1 < n$.

6.3 SVM for Kernel Matrix Completion

In this section, we explore the possibility of using the SVM for problems described in Section 6.2. Notice that the essence of the *em* algorithm lies in (6.4); by using the Frobenius norm instead of Stein's loss, we can modify (6.4) into the following problem:

$$\begin{aligned}
\underset{K_{12},K_{22},w,t}{\text{minimize}} \quad & t \\
\text{subject to} \quad & \left\| \begin{bmatrix} K_{11} & K_{12} \\ K_{12}^T & K_{22} \end{bmatrix} - \sum_{i=1}^{n} w_i M_i \right\|_F \leq t, \\
& \begin{bmatrix} K_{11} & K_{12} \\ K_{12}^T & K_{22} \end{bmatrix} \succeq 0, \\
& w \geq 0,
\end{aligned} \tag{6.5}$$

which is a convex conic program. Compared with (6.4), (6.5) no longer has the full-rank constraint implicitly enforced through the definitions of $\mathcal{K}$ and $\mathcal{M}$. However, like the *em* algorithm, to complete the kernel matrix K for classification problems by solving (6.5) does not take the known class labels of the training data into account. One way to do this is

to incorporate (6.5) into the kernel learning idea proposed in Section 5.1 so that we can complete the kernel matrix K and train an SVM simultaneously. Specifically, for inductive learning, we can formulate the following problem:

$$\begin{aligned} \underset{c,b,\xi,K,w}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta c^T K c + \gamma \left\| K - \sum_{i=1}^{n} w_i M_i \right\|_F \\ \text{subject to} \quad & \operatorname{diag}(y)(Kc + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \quad w \geq 0, \\ & K \succeq 0, \\ & (K)_{ij} = (K_{11})_{ij}, \quad i,j = 1,\ldots,n_1, \end{aligned} \tag{6.6}$$

with variables $c \in \mathbb{R}^n$, $b \in \mathbb{R}$, $\xi \in \mathbb{R}^n$, $K \in \mathbb{R}^{n\times n}$ and $w \in \mathbb{R}^n$. Compared with (5.2), (6.6) replaces the regularization term $\|K - S\|_F$ with $\|K - \sum_{i=1}^{n} w_i M_i\|_F$ in its objective function, and adds constraints to reflect our knowledge of the known entries in K. As detailed in Section 5.1, by letting $z = Kc$, we can rewrite (6.6) as the following convex conic program:

$$\begin{aligned} \underset{z,b,\xi,K,w,u,v}{\text{minimize}} \quad & \frac{1}{n}\mathbf{1}^T\xi + \eta u + \gamma v \\ \text{subject to} \quad & \operatorname{diag}(y)(z + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \quad w \geq 0, \\ & \begin{bmatrix} K & z \\ z^T & u \end{bmatrix} \succeq 0, \\ & \left\| K - \sum_{i=1}^{n} w_i M_i \right\|_F \leq v, \\ & (K)_{ij} = (K_{11})_{ij}, \quad i,j = 1,\ldots,n_1, \end{aligned} \tag{6.7}$$

where $u, v \in \mathbb{R}$ are slack variables.

As specified in Section 6.2 for inductive learning, we are only guaranteed to have auxiliary kernel values $K_{\text{aux}}(x,x)$ and $K_{\text{aux}}(x,x_i)$, $i = 1,\ldots,n$, for a test sample x. Therefore, in order to apply the SVM trained by solving (6.7), we need to infer for x the kernel values

$K(x, x_i)$, $i = 1, \ldots, n$. We first form

$$K'_{\text{aux}} = \begin{bmatrix} & & & K_{\text{aux}}(x, x_1) \\ & K_{\text{aux}} & & \vdots \\ & & & K_{\text{aux}}(x, x_n) \\ K_{\text{aux}}(x, x_1) & \cdots & K_{\text{aux}}(x, x_n) & K_{\text{aux}}(x, x) \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}.$$

Then by performing eigenvalue decomposition[5] on K'_{aux}, we have its eigenvectors β_i, $i = 1, \ldots, n+1$, and we let $M'_i = \beta_i \beta_i^T$, $i = 1, \ldots, n+1$. As a heuristic, we can try to infer $K(x, x_i)$, $i = 1, \ldots, n$, by solving

$$\begin{aligned} \underset{r,t,w}{\text{minimize}} \quad & \left\| \begin{bmatrix} K^\star & r \\ r^T & t \end{bmatrix} - \sum_{i=1}^{n+1} w_i M'_i \right\|_F \\ \text{subject to} \quad & \begin{bmatrix} K^\star & r \\ r^T & t \end{bmatrix} \succeq 0, \end{aligned} \tag{6.8}$$

where $K^\star \in \mathbb{R}^{n\times n}$ is the kernel matrix completed by solving (6.7), and we will use the optimal $r^\star$ as $\begin{bmatrix} K(x, x_1) & \ldots & K(x, x_n) \end{bmatrix}^T$ to classify x. As detailed in Section 5.1.2, we can rewrite (6.8) as the following QCQP:

$$\begin{aligned} \underset{r,t,w}{\text{minimize}} \quad & \left\| \begin{bmatrix} K^\star & r \\ r^T & t \end{bmatrix} - \sum_{i=1}^{n+1} w_i M'_i \right\|_F^2 \\ \text{subject to} \quad & r^T K^\star r - t \leq 0, \\ & \left(I - K^\star (K^\star)^\dagger \right) r = 0, \end{aligned}$$

which is easier to solve than (6.8). If we happen to know $K(x, x)$ and $K(x, x_i)$, $i = 1, \ldots, n_1$, we can still solve (6.8) to infer $K(x, x_i)$, $i = n_1 + 1, \ldots, n$, except that only the unknown kernel values are variables.

[5]Using the procedure proposed in [104], we can do eigenvalue decomposition on K'_{aux} very efficiently by reusing the results of the eigenvalue decomposition on K_{aux}.

Below we also give a variant of (6.6) for transductive learning:

$$\begin{array}{ll} \underset{c,b,\xi,K,w}{\text{minimize}} & \frac{1}{|\mathcal{T}|}\mathbf{1}^T\xi + \eta c^T K_{\mathcal{T}} c + \gamma \left\| K - \sum_{i=1}^{n} w_i M_i \right\|_F \\ \text{subject to} & \operatorname{diag}(y)(K_{\mathcal{T}} c + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \quad w \geq 0, \\ & K \succeq 0, \\ & (K)_{ij} = (K_{11})_{ij}, \quad i,j = 1,\ldots,n_1, \end{array} \tag{6.9}$$

where $K_{\mathcal{T}} \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}$ is the submatrix of K that corresponds to the training set, and here we have $c, \xi \in \mathbb{R}^{|\mathcal{T}|}$. Similarly, we can rewrite (6.9) as the following convex conic program:

$$\begin{array}{ll} \underset{z,b,\xi,K,w,u,v}{\text{minimize}} & \frac{1}{|\mathcal{T}|}\mathbf{1}^T\xi + \eta u + \gamma v \\ \text{subject to} & \operatorname{diag}(y)(z + b\mathbf{1}) \geq \mathbf{1} - \xi, \\ & \xi \geq 0, \quad w \geq 0, \\ & \begin{bmatrix} K_{\mathcal{T}} & z \\ z^T & u \end{bmatrix} \succeq 0, \quad K \succeq 0, \\ & \left\| K - \sum_{i=1}^{n} w_i M_i \right\|_F \leq v, \\ & (K)_{ij} = (K_{11})_{ij}, \quad i,j = 1,\ldots,n_1. \end{array} \tag{6.10}$$

By solving (6.10), we can have the optimal $c^\star$, $b^\star$ and $K^\star$. Then for each test sample x_i, $i \in \{1,\ldots,n\}\backslash\mathcal{T}$, we classify it as

$$\hat{y}_i = \operatorname{sgn}\left(\sum_{j\in\mathcal{T}} c_j^\star K^\star(x_i, x_j) + b^\star \right).$$

Chapter 7
CONCLUSIONS

I have approximate answers and possible beliefs and different degrees of certainty about different things, but I'm not absolutely sure of anything and there are many things I don't know anything about.
Richard Feynman

Similarity-based learning is a practical learning framework for many problems in real applications. Kernel methods can be applied in this framework, but similarity-based learning creates a richer set of challenges because the data may not be natively PSD.

In this dissertation, we explored four different PSD approximations of indefinite similarities: clipping, flipping and shifting the spectrum, and in some cases a pseudoinverse solution. Experimental results show small but sometimes statistically significant differences. Based on the theoretical justification and experimental results, I suggest practitioners spectrum clip. Flipping the spectrum does help achieve significantly better performance for the original Protein problem because, as noted earlier, flipping the spectrum has a similar effect to using the similarities as features, which works favorably for the original Protein problem. However, it should be easy to recognize when spectrum flip will be advantageous, modify the similarity as we did to create the Protein RBF data set, and possibly achieve even better results. Concerning modifying similarities, we addressed the issue of consistent treatment of training and test samples when modifying their similarities to be PSD. Although our proposed solution using matrix multiplication is consistent, we do not contend that it is optimal, and consider this issue still open.

For similarity-based weighted nearest-neighbor classifiers, I first proposed two design goals, namely, affinity and diversity, for the weights, and then proposed two methods for constructing weights that satisfy these two design goals. Experimental results show that

on some real data sets, the proposed weighted k-NN methods can significantly reduce classification error compared to standard k-NN with uniform weights. In particular, on the Caltech-101 data set, which is a benchmark data set for object recognition, the proposed KRI and KRR weights provided a roughly 25% improvement over k-NN with uniform and affinity weights. This clearly demonstrates the importance of considering the design goal of diversity for certain data sets, especially those likely to have highly correlated information. Moreover, according to the empirical evaluation performed on five real data sets, the proposed KRI weights consistently achieve better posterior probability estimates than both the uniform and affinity weights. This implies that the additional cost of solving a QP with k variables incurred by the KRI weights might be worthwhile if the system requires posterior estimates instead of estimated class labels.

Overall, the comparative study shows that local methods are effective for similarity-based learning. It is tempting from the conspicuous discrepancy between the performance of local and global methods on the Amazon-47 and Patrol data sets to conclude that local methods work better than global methods for sparse similarities. However, the Mirex07 data set is also relatively sparse, yet the best methods are global. The Amazon-47 and Patrol data sets are different from the other data sets in that they are the only two data sets with non-maximal self-similarities, and this combined with their sparsity makes the spectra of their similarity matrices have relative large negative components. Therefore, we conjecture that one explanation for the discrepancy is the distortion introduced by the PSD approximation: global methods approximate the indefinite similarity matrix by a PSD matrix on a global scale and seem to introduce more distortion to the original similarities than local methods, which perform the PSD approximation only on a local scale.

For learning from indefinite similarities, I framed the problem as finding a surrogate RKHS, and investigated two methods to simultaneously learn the kernel matrix and minimize the empirical risk with regularization. Experimental evidence suggests that learning a spectrum modification provides an effective trade-off between increased model flexibility and the risk of overfitting. I consider it worthwhile to investigate other forms of regularizers for learning the kernel matrix from indefinite similarities. I showed that these kernel learning ideas can be formulated as convex optimization problems. Furthermore, I

showed that the SVM that learns the spectrum modification, termed the SimSVM, can be solved efficiently by reformulating its optimization problem as an SOCP. I also extended the idea of learning spectrum modification to multiple kernel learning, where indefinite similarities are combined with multiple kernels for the learning task. The focus here was on the SVM, but I hypothesize that this research might also be useful for other types of kernel methods.

For practitioners, the computational cost of a learning algorithm is always a big concern. The SOCP formulation is an efficient implementation of the SimSVM, but it is still more expensive than the standard SVM; according to what we observed during the experiments, on the same data set, the training time for the SimSVM was usually 2-4 times as much as that for the standard SVM. Besides, the SimSVM has two regularization parameters to cross-validate while the standard SVM only has one; however, since cross-validation is completely parallelizable, this should not be a problem if a cluster is used. From the experimental results reported in Section 5.3.2, we can tentatively conclude that for data sets whose spectra have very small negative components such as the Voting data set, the SimSVM is not likely to help; otherwise, the extra computational cost spent on training a SimSVM will very likely result in better classification performance.

Lastly, I discussed the learning problems where we are provided an incomplete kernel matrix with missing rows and columns and also a complete auxiliary kernel matrix. The incomplete kernel matrix, if complete, is assumed to be more informative with regard to the learning task than the auxiliary kernel matrix. Such problems can occur, for example, when we have two views of the samples, yet the view that is more pertinent to the learning task is more difficult to obtain, and therefore we only have it for a subset of the training samples. After reviewing the prior work on completing the kernel matrix with missing rows and columns using auxiliary information, I sketched a direction for future research, namely, a method that extends the kernel learning idea to jointly training an SVM and completing the kernel matrix. Other directions include exploring generative models that model the relationship between the missing information and the auxiliary information more naturally than the one proposed in [109].

BIBLIOGRAPHY

[1] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, January 2003.

[2] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[3] Erling D. Andersen, Cees Roos, and Tamás Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming*, 95(2):249–277, February 2003.

[4] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007. `http://archive.ics.uci.edu/ml/`.

[5] Francis R. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, June 2008.

[6] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21th International Conference on Machine Learning*, 2004.

[7] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. Improved guarantees for learning via similarity functions. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

[8] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1–2):89–112, August 2008.

[9] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, November 2002.

[10] Heinz H. Bauschke, Patrick L. Combettes, and Dominikus Noll. Joint minimization with alternating Bregman proximity operators. *Pacific Journal of Optimization*, 2(3):401–424, September 2006.

[11] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[12] Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, New York, NY, 2nd edition, 2005.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[14] Emamnuel J. Candès and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, June 2010.

[15] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, December 2008.

[16] Luca Cazzanti and Maya R. Gupta. Local similarity discriminant analysis. In *Proceedings of the 24th International Conference on Machine Learning*, pages 137–144, 2007.

[17] Luca Cazzanti and Maya R. Gupta. Regularizing the local similarity discriminant analysis classifier. In *Proceedings of the 8th International Conference on Machine Learning and Applications*, pages 184–189, 2009.

[18] Luca Cazzanti, Maya R. Gupta, and Anjali J. Koppal. Generative models for similarity-based classification. *Pattern Recognition*, 41(7):2289–2297, July 2008.

[19] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[20] Jianhui Chen and Jieping Ye. Training SVM with indefinite kernels. In *Proceedings of the 25th International Conference on Machine Learning*, pages 136–143, 2008.

[21] Yihua Chen, Eric K. Garcia, Maya R. Gupta, Ali Rahimi, and Luca Cazzanti. Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, 10:747–776, March 2009.

[22] Yihua Chen and Maya R. Gupta. Fusing similarities and kernels for classification. In *Proceedings of the 12th International Conference on Information Fusion*, 2009.

[23] Yihua Chen, Maya R. Gupta, and Benjamin Recht. Learning kernels from indefinite similarities. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[24] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[25] J. E. Driskell and T. McDonald. Identification of incomplete networks. Technical report, Florida Maxima Corporation, 2008.

[26] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2nd edition, 2001.

[27] Maryam Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, March 2002.

[28] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In *Proceedings of the 17th IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[29] Shuo Feng, Hamid Krim, and Irina A. Kogan. 3D face recognition using Euclidean integral invariants signature. In *Proceedings of the IEEE Workshop on Statistical Signal Processing*, 2007.

[30] Michael P. Friedlander and Maya R. Gupta. On minimizing distortion and relative entropy. *IEEE Transactions on Information Theory*, 52(1):238–245, January 2006.

[31] Simona Gandrabur, George Foster, and Guy Lapalme. Confidence estimation for NLP applications. *ACM Transactions on Speech and Language Processing*, 3(3):1–29, October 2006.

[32] Itamar Gati and Amos Tversky. Representations of qualitative and quantitative dimensions. *Journal of Experimental Psychology: Human Perception & Performance*, 8(2):325–340, April 1982.

[33] Itamar Gati and Amos Tversky. Weighting common and distinctive features in perceptual and conceptual judgments. *Cognitive Psychology*, 16(3):341–370, July 1984.

[34] Donald Goldfarb, Shucheng Liu, and Siyun Wang. A logarithmic barrier function algorithm for quadratically constrained convex quadratic programming. *SIAM Journal on Optimization*, 1(2):252–267, May 1991.

[35] Robert L. Goldstone and Alan Kersten. Concepts and categorization. In *Comprehensive Handbook of Psychology*, volume 4, chapter 22, pages 599–621. Wiley, New Jersey, 2003.

[36] Mehmet Gönen and Ethem Alpaydm. Localized multiple kernel learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[37] Thore Graepel. Kernel matrix completion by semidefinite programming. In *Proceedings of the International Conference on Artifical Neural Networks*, 2002.

[38] Thore Graepel, Ralf Herbrich, Peter Bollmann-Sdorra, and Klaus Obermayer. Classification on pairwise proximity data. In *Advances in Neural Information Processing Systems*, volume 11, pages 438–444, 1998.

[39] Thore Graepel, Ralf Herbrich, Bernhard Schölkopf, Alex Smola, Peter Bartlett, Klaus-Robert Müller, Klaus Obermayer, and Robert Williamson. Classification on proximity data with LP–machines. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, volume 1, pages 304–309, 1999.

[40] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8:725–760, April 2007.

[41] Arjun K. Gupta and Daya K. Nagar. *Matrix Variate Distributions*. CRC Press, Boca Raton, FL, 1999.

[42] Maya R. Gupta, Robert M. Gray, and Richard A. Olshen. Nonparametric supervised learning by linear interpolation with maximum entropy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):766–781, May 2006.

[43] Bernard Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, April 2005.

[44] Nicholas J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, May 1988.

[45] Sepp Hochreiter and Klaus Obermayer. Support vector machines for dyadic data. *Neural Computation*, 18(6):1472–1510, June 2006.

[46] Thomas Hofmann and Joachim M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, January 1997.

[47] Roger A. Horn and Fuzhen Zhang. Basic properties of the Schur complement. In *The Schur Complement and Its Applications*. Springer, 2005.

[48] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, March 2002.

[49] Tsuyoshi Kato, Koji Tsuda, and Kiyoshi Asai. Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, 21(10):2488–2495, May 2005.

[50] Taishin Kin, Tsuyoshi Kato, and Koji Tsuda. Protein classification via kernel matrix completion. In Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors, *Kernel Methods in Computational Biology*, chapter 12, pages 261–274. MIT Press, Cambridge, MA, 2004.

[51] Ariel Kleiner, Ali Rahimi, and Michael I. Jordan. Random conic pursuit for semidefinite programming. *Submitted for publication*.

[52] Tilman Knebel, Sepp Hochreiter, and Klaus Obermayer. An SMO algorithm for the potential support vector machine. *Neural Computation*, 20(1):271–287, January 2008.

[53] Risi Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

[54] Gert R. G. Lanckriet, Tijl De Bie, Nello Cristianini, Michael I. Jordan, and William S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, November 2004.

[55] Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, January 2004.

[56] Gert R. G. Lanckriet, Minghua Deng, Nello Cristianini, Michael I. Jordan, and William S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 300–311, 2004.

[57] Julian Laub and Klaus-Robert Müller. Feature discovery in non-metric pairwise data. *Journal of Machine Learning Research*, 5:801–808, July 2004.

[58] Julian Laub, Volker Roth, Joachim M. Buhmann, and Klaus-Robert Müller. On the information and representation of non-Euclidean pairwise data. *Pattern Recognition*, 39(10):1815–1826, October 2006.

[59] Li Liao and William S. Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6):857–868, 2003.

[60] Hsuan-Tien Lin and Chih-Jen Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, National Taiwan University, March 2003.

[61] David J. Lipman and William R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, March 1985.

[62] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, November 1998.

[63] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[64] Ronny Luss and Alexandre d'Aspremont. Support vector machine classification with indefinite kernels. In *Advances in Neural Information Processing Systems*, volume 20, pages 953–960, 2007.

[65] Ronny Luss and Alexandre d'Aspremont. Support vector machine classification with indefinite kernels. *Mathematical Programming Computation*, 1(2):97–118, October 2009.

[66] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[67] Sanjay Mehrotra and Jie Sun. A method of analytic centers for quadratically constrained convex quadratic programs. *SIAM Journal on Numerical Analysis*, 28(2):529–544, April 1991.

[68] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, 2000.

[69] Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, July 2005.

[70] Arkadi Nemirovski. Advances in convex optimization: Conic programming. In *Proceedings of the International Congress of Mathematicians*, volume 1, pages 413–444, 2006.

[71] Arkadii Nemirovskii and Katya Scheinberg. Extension of Karmarkar's algorithm onto convex quadratically constrained quadratic problems. *Mathematical Programming*, 72(3):273–289, March 1996.

[72] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, May 2005.

[73] Cheng Soon Ong, Xavier Mary, Stéphane Canu, and Alexander J. Smola. Learning with non-positive kernels. In *Proceedings of the 21st International Conference on Machine Learning*, pages 81–88, 2004.

[74] Elzbieta Pekalska and Robert P. W. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23(8):943–956, June 2002.

[75] Elzbieta Pekalska, Pavel Paclík, and Robert P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, 2:175–211, December 2001.

[76] Scott Philips, James Pitton, and Les Atlas. Perceptual feature identification for active sonar echoes. In *Proceedings of the IEEE Oceans Conference*, 2006.

[77] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–208. MIT Press, 1998.

[78] John C. Platt. Using analytic QP and sparseness to speed training of support vector machines. In *Advances in Neural Information Processing Systems*, volume 11, 1998.

[79] Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, November 2008.

[80] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, January 2004.

[81] Ryan M. Rifkin. *Everything old is new again: A fresh look at historical approaches in machine learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

[82] Volker Roth, Julian Laub, Motoaki Kawanabe, and Joachim M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1540–1551, December 2003.

[83] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, November 2000.

[84] Peter Sadowski, Luca Cazzanti, and Maya R. Gupta. Bayesian and pairwise local similarity discriminant analysis. In *Proceedings of the 2nd International Workshop on Cognitive Information Processing*, 2010.

[85] Simone Santini and Ramesh Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, September 1999.

[86] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, July 2001.

[87] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[88] Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.

[89] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[90] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[91] Alexander J. Smola and Risi Kondor. Kernels and regularization on graphs. In *Proceedings of the 16th Annual Conference on Learning Theory*, 2003.

[92] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, July 2006.

[93] Craig Stanfill and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, December 1986.

[94] Jos F. Strum. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11:625–653, 1999.

[95] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 58(1):267–288, 1996.

[96] Koji Tsuda, Shotaro Akaho, and Kiyoshi Asai. The em algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, May 2003.

[97] Reha H. Tütüncü, Kim-Chuan Toh, and Michael J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95(2):189–217, February 2003.

[98] Amos Tversky. Features of similarity. *Psychological Review*, 84(2):327–352, July 1977.

[99] Amos Tversky and Itamar Gati. Similarity, separability, and the triangle inequality. *Psychological Review*, 89(2):123–154, March 1982.

[100] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, March 1996.

[101] Liwei Wang, Cheng Yang, and Jufu Feng. On learning with dissimilarity functions. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[102] Daphna Weinshall, David W. Jacobs, and Yoram Gdalyahu. Classification in non-metric spaces. In *Advances in Neural Information Processing Systems*, volume 11, pages 838–844, 1998.

[103] Kamil Wnuk and Stefano Soatto. Filtering Internet image search results towards keyword based category recognition. In *Proceedings of the 21st IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[104] Gang Wu, Edward Y. Chang, and Zhihua Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. Technical report, University of California, Santa Barbara, March 2005.

[105] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[106] Yiming Ying, Colin Campbell, and Mark Girolami. Analysis of SVM with indefinite kernels. In *Advances in Neural Information Processing Systems*, volume 22, 2009.

[107] Ming Yuan. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 68(1):49–67, February 2006.

[108] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the 19th IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2126–2136, 2006.

[109] Zhihua Zhang, James T. Kwok, and Dit-Yan Yeung. Model-based transductive learning of the kernel matrix. *Machine Learning*, 63(1):69–101, April 2006.

[110] Xiaojin Zhu and Andrew B. Goldberg. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2009.

[111] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[112] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 67(2):301–320, April 2005.

Appendix A

PROOF OF PROPOSITION 4.1

Proof. Recall the eigenvalue decomposition $S_{\text{clip}} = U\Lambda_{\text{clip}}U^T$. For $\Lambda_{\text{clip}} = \text{diag}(\lambda_1, \ldots, \lambda_n)$, assume that $\lambda_1 \geq \ldots \geq \lambda_m > 0$ and $\lambda_{m+1} = \ldots = \lambda_n = 0$. Let

$$D = \text{diag}(\lambda_1, \ldots, \lambda_m) \in \mathbb{R}^{m \times m},$$

and

$$V = \begin{bmatrix} u_1 & \ldots & u_m \end{bmatrix} \in \mathbb{R}^{n \times m},$$

where u_i is the ith column of U, then we have $S_{\text{clip}} = VDV^T$. The vector representation of the training samples in $\mathbb{R}^m$ implied by S_{clip} is

$$X = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix} = D^{1/2}V^T \in \mathbb{R}^{m \times n}.$$

Given a test similarity vector $s \in \mathbb{R}^n$, the least-squares solution to the equation $X^T x = s$ is $x = (XX^T)^{-1}Xs$. Let $\tilde{s}$ be the vector of the inner products between the embedded test sample x and the embedded training samples X, then based on the fact that $V^TV = I_m$, where I_m denotes the $m \times m$ identity matrix, we have

$$\begin{aligned}
\tilde{s} &= X^T x \\
&= X^T(XX^T)^{-1}Xs \\
&= VD^{1/2}\left(D^{1/2}V^TVD^{1/2}\right)^{-1}D^{1/2}V^Ts \\
&= VD^{1/2}D^{-1}D^{1/2}V^Ts \\
&= VV^Ts \\
&= UD_{\text{clip}}U^Ts \\
&= P_{\text{clip}}s.
\end{aligned}$$

This ends the proof. □

Appendix B

PROOF OF LEMMA 5.3

Proof. For $y \geq 0$ and $z \geq 0$, we have

$$\left\| \begin{bmatrix} 2x \\ y - z \end{bmatrix} \right\|_2 \leq y + z \;\Leftrightarrow\; \left\| \begin{bmatrix} 2x \\ y - z \end{bmatrix} \right\|_2^2 \leq (y + z)^2.$$

Since

$$\left\| \begin{bmatrix} 2x \\ y - z \end{bmatrix} \right\|_2^2 = \begin{bmatrix} 2x^T & y - z \end{bmatrix} \begin{bmatrix} 2x \\ y - z \end{bmatrix} = 4x^T x + y^2 - 2yz + z^2,$$

and

$$4x^T x + y^2 - 2yz + z^2 \leq (y + z)^2 \;\Leftrightarrow\; x^T x \leq yz,$$

we can conclude that for $y \geq 0$ and $z \geq 0$, we have

$$\left\| \begin{bmatrix} 2x \\ y - z \end{bmatrix} \right\|_2 \leq y + z \;\Leftrightarrow\; x^T x \leq yz.$$

This ends the proof. □

Geometrically, for $x \in \mathbb{R}^n$ and $y, z \in R$, the following restricted hyperbolic constraints[1]

$$x^T x \leq 2yz, \quad y \geq 0, \quad z \geq 0, \tag{B.1}$$

represent a second-order cone rotated 45 degrees in the (y, z)-plane. To see this, we know by Lemma 5.3 that (B.1) is equivalent to

$$\left\| \begin{bmatrix} x \\ \frac{1}{\sqrt{2}} y - \frac{1}{\sqrt{2}} z \end{bmatrix} \right\|_2 \leq \frac{1}{\sqrt{2}} y + \frac{1}{\sqrt{2}} z, \quad y \geq 0, \quad z \geq 0. \tag{B.2}$$

[1]The constraints in (B.1) are slightly different from those stated in Lemma 5.3 in that here we have $x^T x \leq 2yz$ instead of $x^T x \leq yz$.

Now we just need to rotate it back by 45 degrees in the (y, z)-plane:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} I_n & 0 & 0 \\ 0 & \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} I_n & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{B.3}$$

where I_n is the $n \times n$ identity matrix. By substituting (B.3) into (B.2), we obtain

$$\left\| \begin{bmatrix} x' \\ y' \end{bmatrix} \right\|_2 \leq z',$$

which represents a second-order cone in $\mathbb{R}^{n+2}$.

Appendix C

WISHART AND INVERSE WISHART DISTRIBUTIONS

Definition C.1 (Wishart Distribution)**.** A $p \times p$ random symmetric positive definite matrix S is said to have a Wishart distribution with parameters p, ν and Σ for $\nu > p-1, \Sigma \in \mathbb{R}^{p\times p}$ and $\Sigma \succ 0$, written as $S \sim \mathcal{W}_p(\nu, \Sigma)$, if its probability density function (p.d.f.) is given by

$$p(S) = \frac{(\det S)^{\frac{1}{2}(\nu-p-1)}}{2^{\frac{1}{2}\nu p}\Gamma_p\left(\frac{1}{2}\nu\right)(\det \Sigma)^{\frac{1}{2}\nu}} \exp\left(-\frac{1}{2}\operatorname{tr}\left(\Sigma^{-1}S\right)\right),$$

where $\Gamma_p(\cdot)$ is the multivariate gamma function defined as

$$\Gamma_p(x) = \pi^{\frac{1}{4}p(p-1)} \prod_{i=1}^{p} \Gamma\left(x - \frac{1}{2}(i-1)\right).$$

Theorem C.2. *Let $S \sim \mathcal{W}_p(\nu, \Sigma)$, then $E(S) = \nu\Sigma$, and $\operatorname{Cov}(S_{ij}, S_{kl}) = \nu(\Sigma_{ik}\Sigma_{jl} + \Sigma_{il}\Sigma_{jk})$.*

Theorem C.3. *Let $X_1, \ldots, X_n \in \mathbb{R}^p$ be i.i.d. random vectors drawn from a multivariate Gaussian distribution $\mathcal{N}_p(0, \Sigma)$ with $n \geq p$. Let $X = \begin{bmatrix} X_1 & \ldots & X_n \end{bmatrix} \in \mathbb{R}^{p\times n}$, then $XX^T \sim \mathcal{W}_p(n, \Sigma)$.*

Theorem C.4. *Let $X_1, \ldots, X_n \in \mathbb{R}^p$ be i.i.d. random vectors drawn from $\mathcal{N}_p(\mu, \Sigma)$ with $n > p$. Define $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $S = \sum_{i=1}^{n}(X_i - \hat{\mu})(X_i - \hat{\mu})^T$. Then (i) $\hat{\mu}$ and S are independently distributed, (ii) $\hat{\mu} \sim \mathcal{N}_p(\mu, \frac{1}{n}\Sigma)$, and (iii) $S \sim \mathcal{W}_p(n-1, \Sigma)$.*

Theorem C.5. *Let $S \sim \mathcal{W}_p(\nu, \Sigma)$. Then for any $A \in \mathbb{R}^{q\times p}$ with $\operatorname{rank}(A) = q \leq p$, we have $ASA^T \sim \mathcal{W}_q(\nu, A\Sigma A^T)$.*

Theorem C.6. *Let $S_1, \ldots, S_k$ be independently distributed with $S_j \sim \mathcal{W}_p(\nu_j, \Sigma)$, $j = 1, \ldots, k$. Then $\sum_{j=1}^{k} S_j \sim \mathcal{W}_p(\sum_{j=1}^{k}\nu_j, \Sigma)$.*

Definition C.7 (Inverse Wishart Distribution)**.** A $p \times p$ random symmetric positive definite matrix V is said to have an inverse Wishart distribution (also called inverted Wishart distribution) with parameters p, ν and Ψ for $\nu > p-1$, $\Psi \in \mathbb{R}^{p\times p}$ and $\Psi \succ 0$, written as $V \sim \mathcal{IW}_p(\nu, \Psi)$, if its p.d.f. is given by

$$p(V) = \frac{(\det \Psi)^{\frac{1}{2}\nu}}{2^{\frac{1}{2}\nu p}\Gamma_p\left(\frac{1}{2}\nu\right)(\det V)^{\frac{1}{2}(\nu+p+1)}} \exp\left(-\frac{1}{2}\operatorname{tr}\left(V^{-1}\Psi\right)\right).$$

Theorem C.8. *Let* $V \sim \mathcal{IW}_p(\nu, \Psi)$ *with* $\nu > p + 1$, *then* $E(V) = \frac{1}{\nu - p - 1}\Psi$.

Theorem C.9. *Let* $S \sim \mathcal{W}_p(\nu, \Sigma)$, *then* $S^{-1} \sim \mathcal{IW}_p(\nu, \Sigma^{-1})$. *Similarly, let* $V \sim \mathcal{IW}_p(\nu, \Psi)$, *then* $V^{-1} \sim \mathcal{W}_p(\nu, \Psi^{-1})$.

VITA

Yihua Chen received the B.Sc. and M.Sc. degrees both in electrical engineering from Shanghai Jiao Tong University, China, in 2003 and 2006. He joined the Electrical Engineering Department of the University of Washington in 2006, and has since been working with Prof. Maya R. Gupta on statistical learning problems. He has also interned at Microsoft Research Redmond, Microsoft Research Asia and Windows Live China.