ABSTRACT

Title of dissertation: LEARNING WITH MULTIPLE SIMILARITIES
Abhishek Kumar, Doctor of Philosophy, 2013

Dissertation directed by: Professor Hal Daumé III
Department of Computer Science

The notion of similarities between data points is central to many classification and clustering algorithms. We often encounter situations when there are more than one set of pairwise similarity graphs between objects, either arising from different measures of similarity between objects or from a single similarity measure defined on multiple data representations, or a combination of these. Such examples can be found in various applications in computer vision, natural language processing and computational biology. Combining information from these multiple sources is often beneficial in learning meaningful concepts from data.

This dissertation proposes novel methods to effectively fuse information from these multiple similarity graphs, targeted towards two fundamental tasks in machine learning - classification and clustering. In particular, I propose two models for learning spectral embedding from multiple similarity graphs using ideas from co-training and co-regularization. Further, I propose a novel approach to the problem of multiple kernel learning (MKL), converting it to a more familiar problem of binary classification in a transformed space. The proposed MKL approach learns a "good" linear combination of base kernels by optimizing a quality criterion that is justified both empirically and theoretically. The ideas of the proposed MKL method are also extended to learning nonlinear combinations of kernels, in particular, polynomial kernel combination and more general nonlinear kernel combination using random forests.

Learning with Multiple Similarities

by

Abhishek Kumar

Dissertation presented to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:

Professor Hal Daumé III, Chair/Adviser
Professor Ramani Duraiswami
Professor Lise Getoor
Professor David W. Jacobs
Professor P. S. Krishnaprasad

To my parents and my wife.

Acknowledgments

I consider myself extremely fortunate to have Hal Daumé III as my advisor. Hal has given me enough freedom to explore different problems and pursue my own research interests, and I am grateful to him for having faith in my abilities. It was always a pleasure stepping into his office and have immediate access to his immense breadth of knowledge, thorough technical expertise and sharp intuitions that would turn out to be crucial in shaping up my thoughts for subsequent research. The interactions with him always left me filled with positive energy. I thank him for being a great advisor throughout my graduate studies – I could not have asked for anyone better.

I thank Professors Ramani Duraiswami, Lise Getoor, David Jacobs and P. S. Krishnaprasad for serving on my Ph.D. committee, asking insightful questions and providing helpful feedback. I have learned a great deal from a few people during my graduate studies who I want to thank here. I had the absolute pleasure of working with Thomas P. Fletcher in my first year of Ph.D., and whatever little I know of matrix manifolds, I owe it to Tom. I greatly admire his style of teaching and some of the best talks, I have ever been to, are by him. I also had the privilege of working with John H.L. Hansen during my Masters and I thank him for teaching me Speech Recognition. I had two wonderful internship experiences during my Ph.D. I thank Alexandru Niculescu-Mizil for having me as a summer intern at NEC Labs. It was a delightful learning experience working with Alex and I greatly admire his ability to think deep on a problem and his enthusiasm about research. The work I did there ended up being part of this thesis. I thank Vikas Sindhwani for having me as a summer intern at IBM Research, for many thought-provoking discussions that we had, and for his valuable advice on matters related to work and beyond. Vikas has so many qualities I would like to emulate as a researcher.

I would also like to thank Piyush Rai, Avishek Saha and Abhishek Sharma for collaborating with me on different projects. My first paper in machine learning was in collaboration with Avishek and it was a good learning experience. I have turned to Piyush for help and advice on several occasions over the years and I thank him for always being there. It is always a pleasure working with him. I thank Abhishek for numerous lively discussions on topics ranging from highly technical to philosophical, material, political, etc. I appreciate his enthusiasm and eagerness for learning. I would like to acknowledge the help and support of my lab-mates at various points in time, and thank them for many enjoyable conversations. I also thank the administrative and computing staff at Computer Science department and UMIACS, in particular to Jennifer Story, Fatima Bangura, Arlene Schenk, Janice Perrone, Petra Zapf and Joe Webster, for being utterly helpful and accommodating.

I am grateful to Praveen Noojipady and Yuanjie Li for always being there when I needed them. Special thanks to their sweet little daughter Meghana for several fun-filled moments. I will never forget the wonderful gatherings and engaging conversations we had with our other friends, which were a refresher from routine graduate life.

I will fall short of words in expressing my indebtedness to my parents whose love, nurturing and guidance was pivotal in my formative years and enabled me to embark on this journey. Finally, this thesis would not have been possible without constant love and support of my wife, Sumita, who has always encouraged me to pursue my dreams.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Adj-RI	Adjusted Rand Index
CCA	Canonical Correlation Analysis
Fisher-LDA	Fisher Linear Discriminant Analysis
LSA	Latent Semantic Analysis
LWS	Learning with Similarity Functions
MKL	Multiple Kernel Learning
NMI	Normalized Mutual Information
PCA	Principle Component Analysis
RKHS	Reproducing Kernel Hilbert Space
SIFT	Scale Invariant Feature Transform
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TS-MKL	Two Stage Multiple Kernel Learning

Chapter 1

Introduction

Recent few years have witnessed rapid growth in the amount of content and associated data. In many cases, the data representing an entity originates from diverse sources and manifest itself in different *views* or *modalities*. One example of this phenomena is content on `wikipedia`, where a topic or event is described in many different languages. Images on the Internet often have tags or captions associated with them, which represent another *view* of the same underlying content. Other than aforementioned examples where data is naturally present in multiple views, we can also extract different sets of features or define different similarity measures between two data objects which results in multiple views of data. Combining information from these multiple views or modalities is often beneficial in learning meaningful concepts from data. For the scope of this dissertation, we will work with the setting where all the information is given to us in the form of real valued similarities for each pair of data points. If we are given feature representations of individual data examples, it is assumed that appropriate similarity measures can be applied on pairs of data points to give us the similarity representation of data. This dissertation proposes novel methods to effectively fuse information from these multiple similarity views of data, targeted towards two fundamental tasks in machine learning – classification and clustering.

Machine learning, on a broad level, is concerned with learning meaningful concepts from data. The problems arising in the field of machine learning can be broadly classified under two categories: (i) *Supervised learning*, where data is accompanied by concept class labels that can used in guiding the learning process, and (ii) *Unsupervised learning*, where data comes unlabeled and the learning process has to discover meaningful concepts from unlabeled data. Labeled data is usually in the form of a collection of labeled *examples* $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ where $\mathbf{x}_i$ is the feature vector and $\mathbf{y}_i$ is the associated label (scalar or vector). Unlabeled data is a collection of unlabeled examples $(\mathbf{x}_i)_{i=1}^m$. *Classification* is a supervised learning problem in which the training data contains labeled examples with discrete-valued labels and the goal is predict the class labels of future test examples. *Regression* is also a supervised learning problem where the training data consists of input features (explanatory variables) and associated continuous-valued labels (response variables), and the goal is to predict the responses for future test examples. *Clustering* is an unsupervised learning problem in which the learner is given a pool of unlabeled examples and the goal is to learn "meaningful" grouping structure or cluster boundaries to unravel patterns in the data.

1.1 Learning with Similarities

A machine learning algorithm normally uses either labeled examples $((\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n)$ or unlabeled examples $(\mathbf{x}_i)_{i=1}^n$ to learn an underlying concept. A *similarity function* $\mathcal{S} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ takes a

pair of examples (the feature vectors $\mathbf{x}_i$, $\mathbf{x}_j \in \mathbb{R}^d$) as input and maps to a real value that acts as a measure of *similarity* between example pairs. In similarity based learning, the learning algorithm works with similarities between example pairs instead of working with individual example features.

A typical example of similarity based learning is Support Vector Machine (SVM) (Boser et al., 1992; Cortes & Vapnik, 1995) which is a widely used learning algorithm for classification. Nonlinear SVM uses the so-called "kernel trick" to map the example pairs to real valued similarities that are used in learning nonlinear classifier boundaries. The similarity function used in the kernel-trick is a positive semidefinite function $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ referred to as the kernel. The function space induced by a positive semidefinite kernel is called a *Reproducing Kernel Hilbert Space (RKHS)*, and the subject enjoys a rich history in the literature (Aronszajn, 1950). The theory of positive semidefinite kernels and the associated RKHS has been applied to a great success in machine learning, especially in classification problems through SVMs.The notion of similarity is also a basis for many clustering algorithms which aim to put "similar" examples under same cluster and "dissimilar" examples in different clusters (Dhillon et al., 2004; Biehl et al., 2009).

1.2 Multiple Views of an Entity

With the increase in the amount of data, it is becoming more likely to encounter scenarios where we have more than one set of observations for a single entity, also referred as *views*. Below we give some application instances of this scenario.

1.2.1 Examples from Computer Vision

Computer vision is broadly concerned with processing and analyzing visual information, mostly in the form of images or videos, to make decisions or to discover meaningful concepts. Images or videos available on the Internet (e.g., images on `flicker`, videos on `youtube`) are often accompanied with tags and description of the content in text form. In this case, the visual content and the associated text describing it are the two *views* of the same entity or content, both of which can be used in tasks of interest.

It is also common to have a scene captured from different camera angles, e.g., face images of a person in different poses (front, side-left, side-right, etc.). Incorporating all these *views* of a scene in the learning process can lead to better performance than using only a single *view*. One of the many examples is the problem of image segmentation where images from multiple camera angles can be used to improve segmentation of the scene (Xiao & Quan, 2009).

Apart from the above scenarios where the *multiple views* happen to occur more naturally, it is also common in many object recognition problems to extract multiple feature sets from images capturing information at different levels (SIFT features, Gabor features, wavelet features, etc.) and use all of these to improve recognition accuracy (Gehler & Nowozin, 2009). Each feature set can be treated as a *view* of the same underlying content in this case.

1.2.2 Examples from Natural Language Processing

Natural Language Processing (NLP) is broadly concerned with computational understanding of human languages. There has been an explosion in the amount of text data available on the

Internet due to increasing popularity of web-logs, `wikipedia`, etc. and a significant portion is in languages other than English. It is not uncommon to encounter situations where a single topic or event is described in multiple languages. A typical example is articles on `wikipedia` where same topic or event is described in multiple languages. Description in each language can serve as a *view* of the underlying content and information from all these views can be utilized to improve the performance of the tasks of interest (Boyd-Graber & Blei, 2009; Ni et al., 2009; Jagarlamudi & Daumé III, 2010).

It is also possible to have information in multiple views to be organized in different forms, e.g., features of individual examples, relationships between multiple examples, etc. A typical instance of this is data on `World Wide Web` where each web page has the text content as one view (and if applicable, image content as second view) and the incoming-outgoing links between web pages as another view (Blum & Mitchell, 1998). Another instance is found in scientific literature data where text of a publication serves as one view and the past publications that it cites (or future publications that cite it), serve as another view (Mei et al., 2008).

Multi-view examples are also encountered in Bio-informatics where multiple similarity measures can be derived from data pertaining to a single phenomenon (Zien & Ong, 2007; Mostafavi & Morris, 2010).

1.3 An Overview of This Dissertation

This dissertation presents novel methods for the problem of learning with multiple views when data in each view is available as pairwise similarities of examples. The goal is to effectively combine information from multiple views so as to improve the performance of the end task. In particular, all the proposed methods are targeted towards one of the two machine learning tasks – classification and clustering.

Several machine learning methods or algorithms exist in the literature for classification task. The framework of large margin classification and Support Vector Machines (SVM) have been particularly popular with both users and researchers of machine learning. Support Vector Machines offer a principled way to build linear and nonlinear classifiers and also enjoy strong theoretical guarantees. They have been applied in numerous application domains (Moguerza & Munoz, 2006; Yang, 2004; Joachims, 1998; Tay & Cao, 2001) and have also been a founding base for a rich body of follow-on work. Due to its popularity and wide applicability, the methods proposed in this dissertation are designed to be used in conjunction with Support Vector Machine as the data classifier.

Several algorithms have been proposed in the literature for the problem of clustering as well. Compared to traditional algorithms like k-means or single linkage clustering, spectral clustering has many fundamental advantages (von Luxburg, 2007). Besides being easy to implement, it is also theoretically motivated from a graph cut point of view and has efficient implementations using fast linear algebraic methods. Spectral clustering has been widely used in various application domains (Ning et al., 2007; Valgren et al., 2007; Bach & Jordan, 2006; Wang et al., 2012; Ekin et al., 2004). A rich body of literature has been built up around spectral clustering which is evident from the fact that one of the earliest spectral clustering papers in the machine learning literature (Ng et al., 2002) has close to 3000 citations in the last 10 years. The multiview methods proposed in this dissertation for clustering task are specifically targeted to work for spectral clustering, again

due to its wide applicability and popularity.

The rest of this dissertation is organized as the following Chapters.

Chapter 2: This chapter provides a brisk background on spectral clustering, Support Vector Machine (SVM) and the kernel trick, and Multiple Kernel Learning (MKL) that is needed for the subsequent chapters. Relevant previous work is also discussed.

Part I: Spectral Embeddings from Multiple Similarity Graphs

Chapter 3: Co-training was first introduced by Blum & Mitchell (1998) for the problem of semi-supervised learning with multiple views and has enjoyed significant empirical success since then (Nigam & Ghani, 2000; Sarkar, 2001; Müller et al., 2002; Callison-Burch & Osborne, 2003; Chen et al., 2011). However, the potential of co-training has not been explored much in unsupervised learning problems except in a few prior works (de Sa, 2005; Bickel & Scheffer, 2004). In this Chapter, I present a novel co-training based approach to learn spectral embeddings by fusing information from multiple similarity graphs. The learned spectral embedding is evaluated on the problem of clustering and can be seen as a multi-view spectral clustering approach.

Chapter 4: The idea of co-regularization has also been popular in the semi-supervised learning literature (Sindhwani et al., 2005; Daumé III et al., 2010). However it has not been explored for unsupervised learning problems, especially in the context of spectral clustering. This chapter presents an approach based on the idea of co-regularization to learn spectral embeddings from multiple similarity graphs. The learned spectral embedding is again evaluated on the problem of clustering, giving rise to a multi-view spectral clustering method.

Part II: Learning with Multiple Kernels

Chapter 5: Multiple Kernel Learning (MKL) was introduced in Cristianini et al. (2001); Kandola et al. (2002); Lanckriet et al. (2004); Bach et al. (2004) and it has generated significant activity in the machine learning community since then (Rakotomamonjy et al., 2007; Sonnenburg et al., 2006; Cortes et al., 2010a; Kloft et al., 2011; Bach, 2008; Zien & Ong, 2007; Cortes et al., 2009a; Sindhwani & Lozano, 2011). In the MKL problem, we are given labeled data with multiple kernel functions and the goal is to combine these multiple kernels so as to improve the prediction performance. Most of the effort in the area has been on faster optimization methods for the MKL framework proposed in Lanckriet et al. (2004). Unfortunately, MKL has not achieved the expected empirical success in terms of prediction performance. This Chapter presents a novel two stage framework for the MKL problem that transforms the MKL problem into the familiar classification problem in a transformed space. The proposed approach shows better empirical performance despite its simplicity. Theoretical guarantees are also provided.

Chapter 6: This chapter extends the ideas of previous chapter to learn nonlinear kernel combinations. I study two types of kernel combinations: (i) polynomial kernel combinations that result in positive semidefinite kernels, (ii) more general nonlinear kernel combination using random forests that result in indefinite kernel, which is used with learning-with-similarity functions (LWS) framework of Balcan & Blum (2006).

Chapter 7: This chapter summarizes the contributions of the dissertation and outlines directions for future study.

Chapter 2

Background

This chapter introduces the background that will be needed for subsequent chapters. First, I briefly introduce spectral clustering, which forms the basis for Chapters 3 and 4. The subsequent sections give a brief introduction to Support Vector Machines, kernels and Multiple Kernel Learning which will be needed to put Chapters 5 and 6 in right perspective with respect to the existing work.

2.1 Spectral Clustering

Spectral clustering is a technique that exploits the properties of the Laplacian of the graph whose edges denote the similarities between the data points. The bottom k eigenvectors of the normalized graph Laplacian are relaxations of the indicator vectors that assign each node in the graph to one of the k clusters. Apart from being theoretically well-motivated, spectral clustering has the advantage of performing well on arbitrary shaped clusters, which is otherwise a shortcoming with several other clustering algorithms such as the k-means algorithm. Here we briefly outline the spectral clustering algorithm due to Ng et al. (2002):

- Construct an $n \times n$ non-negative symmetric matrix $\mathbf{K}$, where $\mathbf{K}_{ij}$ quantifies the similarity between samples i and j.
- Construct the matrix $\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{K}_{ij}$. The matrix $(I - \mathcal{L})$ is also called normalized graph Laplacian.
- Let $\mathbf{U}$ denote a $n \times k$ matrix with columns as the top k eigenvectors of $\mathcal{L}$
- Normalize each row of $\mathbf{U}$ to obtain $\mathbf{V}$.
- Run the k-means algorithm to cluster the row vectors of $\mathbf{V}$.
- Assign example i to cluster c if the i-th row of $\mathbf{V}$ is assigned to cluster c by the k-means algorithm.

For a detailed introduction to both theoretical and practical aspects of spectral clustering, the reader is referred to the excellent tutorial by von Luxburg (2007).

2.2 Support Vector Machines

In this section, we give a brief introduction to Support Vector Machine (SVM) for classification and the use of kernels for nonlinear classification, which will set the foundation to motivate and introduce the subject of multiple kernel learning (MKL) later in this chapter.

2.2.1 Large Margin Classification

Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$ be the features of training data with $\{y_1, y_2, \ldots, y_n\} \in \{-1, +1\}$ as their corresponding class labels. The goal in classification is to learn a function $f : \mathbb{R}^d \mapsto \mathbb{R}$ that maps input features to a real value whose sign can be used to determine the label of the concerned example. For so-called linear classifiers, f is just an affine function with $f(x) = \mathbf{w}'\mathbf{x} + b$ and vector $\mathbf{w}$ is the normal to the *decision hyperplane* (or *separating hyperplane*). The **margin** of i'th example is defined as $y_i(\mathbf{w}'\mathbf{x}_i + b)$ which indicates the confidence of the classifier on that particular example. The ℓ_p distance of point $\mathbf{x}_i$ from the separating hyperplane is $\frac{|\mathbf{w}'\mathbf{x}_i+b|}{\|\mathbf{w}\|_q}$ where ℓ_p and ℓ_q norms are dual to each other ($1/p + 1/q = 1$). The quantity $\min_i y_i(\mathbf{w}'\mathbf{x}_i + b)$ is the *minimum margin* over all the examples. We say that a classifier is a **large margin classifier** if it maximizes the minimum margin with some norm constraint on $\|\mathbf{w}\|$.

2.2.2 History

The support Vector Machine (SVM) classifier learns a linear hyperplane, either in the original feature space or in a higher dimensional space induced by a *kernel*, that separates the positive and negative examples by maximum margin. The fundamental ideas underlying the present Support Vector Machine (SVM) including the large margin separation, support vectors (referred as *extreme vectors* then) were first introduced in the *Generalized Portrait algorithm* by Vapnik & Lerner (1963); Vapnik & Chervonenkis (1964). Mangasarian (1965) proposed a linear programming approach to large margin classification of linearly separable data by constraining the ℓ_∞ norm of the weight vector. A linear program making use of slack variables to learn a classifier for non-separable data was proposed by Smith (1968). The ideas of large margin classification and controlling the capacity or expressive power of the hypothesis class (i.e., family of functions to which the classifier belongs) to get better *generalization* were formalized in statistical learning theory by Vapnik (1979) and the principle of *structural risk minimization (SRM)* was introduced. Vapnik (1979) showed that the expected error of a classifier can be upper bounded by its training error plus a term depending on the *complexity* of the hypothesis class measured in terms of *VC-dimension*. The SRM principle asked to choose a classifier that minimizes this upper bound striking a right balance between training error and complexity term. It was shown that VC-dimension of linear classifiers with unit norm constrained weight vectors and with minimum margin ρ can be upper bounded by $\min\{R^2/\rho^2, d\} + 1$ where $R = \max_i \|\mathbf{x}_i\|$ (Vapnik, 1979). This justified the idea of large margin classifiers formally from structural risk minimization point of view. The SVM for linearly separable data, as we know it today, was introduced by Boser et al. (1992) which presented the well known quadratic program to solve the SVM in the dual. Cortes & Vapnik (1995) presented soft margin SVM for non-separable data using slack variables.

2.2.3 SVM optimization problem

For linearly separable classes, we want to look for a $\mathbf{w}$ such that $\mathbf{w}'\mathbf{x}_i + b > 0$ for positive class examples and $\mathbf{w}'\mathbf{x}_i + b < 0$ for negative class examples. The ℓ_2 distance of a point $\mathbf{x}_i$ from such a decision hyperplane $\mathbf{w}'\mathbf{x}_i + b = 0$ is given as $\frac{|\mathbf{w}'\mathbf{x}_i+b|}{\|\mathbf{w}\|_2} = \frac{y_i(\mathbf{w}'\mathbf{x}_i+b)}{\|\mathbf{w}\|_2}$. As discussed earlier, large margin classifiers maximize the minimum distance of points from the decision hyperplane, which

gives rise to the following optimization problem:

$$\{\mathbf{w}^*, b^*\} = \arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t. } y_i(\mathbf{w}'\mathbf{x}_i + b) \geq 1 \tag{2.1}$$

The dual problem of the constrained optimization problem of Eq. 2.1 is given as

$$\max_{\boldsymbol{\alpha}\geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j \quad \text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.2}$$

For non-separable classes, we can introduce slack variables $\{\xi_i\}_{i=1}^n$ and relax the margin constraints to $y_i(\mathbf{w}'\mathbf{x}_i+b) \geq 1-\xi_i$ with $\xi_i \geq 0$ for all i. The primal problem now becomes $\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i$ subject to these relaxed margin constraints, where C is a trade-off hyperparameter. Forming the dual for this soft margin problem as above, we get the following optimization problem.

$$\max_{0\leq\boldsymbol{\alpha}\leq C} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j \quad \text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.3}$$

The decision hyperplane is given by $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$, and the predicted class label for a test example $\mathbf{x}$ is given as

$$\hat{y} = \text{sign}(\mathbf{w}'\mathbf{x} + b) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i'\mathbf{x} + b). \tag{2.4}$$

KKT complementary slackness conditions ensure that α_i's are zero for those data points for which the margin constraints are inactive, i.e., $y_i(\mathbf{w}'\mathbf{x}_i + b) > 1$. The rest of the points, for which the margin constraints are active, are called **support vectors** and have their corresponding α_i's positive. The SVM decision hyperplane only depends on these support vectors.

2.2.4 Feature spaces and the kernel trick

If the data is not linearly separable, it might be possible to nonlinearly transform the input features $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ to another, possibly higher dimensional, *feature space* using a transform $\phi : \mathcal{X} \mapsto \mathcal{F}$ and do linear classification there. It can be noticed that in the optimization problem of Eq. 2.3 as well as in the prediction function of Eq. 2.4, the inputs make appearance only in pairwise manner through dot products. This has important consequences as we do not need to compute the transformed features explicitly for using SVMs and it suffices to just compute the dot products in the transformed feature space. The new feature space $\mathcal{F}$ should be a dot product space. For some feature spaces, it is possible to directly evaluate the dot products using a symmetric **kernel function** $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ without explicitly computing the features ($k(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$) (Aizerman et al., 1964; Poggio, 1975; Boser et al., 1992). Such kernels are called **positive semidefinite kernels** or **Mercer kernels** (Mercer, 1909; Aizerman et al., 1964) after James Mercer who proved that kernel of a positive linear operator can be expanded as series sum of products of its eigenfunctions $\{\psi_i(\cdot)\}_{i=1}^E$, i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{e=1}^E \lambda_e \psi_e(\mathbf{x}_i)\psi_e(\mathbf{x}_j)$, where E is the number of positive eigenvalues of the linear operator. One possibility for implicitly mapped feature corresponding to the dot product obtained by this kernel function can be

$\phi(\mathbf{x}_i) = [\sqrt{\lambda_1}\psi_1(\mathbf{x}_i), \sqrt{\lambda_2}\psi_2(\mathbf{x}_i), \ldots, \sqrt{\lambda_E}\psi_E(\mathbf{x}_i)]$. Another possibility is to think of mapped feature as $\phi(\mathbf{x}_i) = k(\mathbf{x}_i, \cdot)$ which is a function. For the kernel to evaluate the dot product in the feature space, we should have $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot) \rangle$. This property of the kernel function is called reproducing kernel property and it holds for Mercer kernels. The linear span of all $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^n$ is called **reproducing kernel Hilbert space (RKHS)** (Aronszajn, 1950).

For any Mercer kernel $k(\cdot, \cdot)$ that induces an implicit feature map $\phi : \mathcal{X} \mapsto \mathcal{F}$, we can write the SVM optimization problem of Eq. 2.3 as

$$\max_{0 \leq \boldsymbol{\alpha} \leq C} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.5}$$

The prediction for an example $\mathbf{x}$ is given by $\hat{y} = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b)$. Some of the popular kernel functions used in practice are Gaussian kernel or radial basis function (RBF) kernel ($k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$), polynomial kernel ($k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i' \mathbf{x}_j + 1)^g$), etc. Occasionally, people also use kernels that do not satisfy Mercer requirement, like Sigmoid "kernel". While these often work well in practice, in theory the use of a indefinite kernel results in non-convexity of the SVM objective. Also, it loses the intuitive interpretation of a dot product in an induced feature space.

For a detailed description of the theory and practice of large margin classification and Support Vector Machines, the reader is referred to (Burges, 1998; Smola et al., 1999).

2.3 Multiple Kernel Learning

A crucial component of the SVM learning is the kernel function that maps the inputs to a feature space where a linear classifier is expected to separate the data. Although SVMs have been applied to many applications with significant empirical success, its performance critically hinges upon the choice of the kernel function, which is usually hard to select even when the user has a good familiarity with the problem domain. For this reason, automated learning of the kernel function has been an active area of research. The majority of the previous work in this area has focused on the Multiple Kernel Learning (MKL) setting, where the user is only tasked with specifying a set of **base kernels**, and the learning algorithm is in charge of finding a combination of these base kernels that is appropriate for the problem at hand. Most of the methods proposed in the literature work with nonnegative linear combination of base kernels to ensure that the final kernel is positive semidefinite. Next, we briefly describe two main lines of work in this area.

2.3.1 One stage MKL

In one stage MKL, both the weights of the linear combination of base kernels and the parameters of the SVM classifier are learned by solving a single joint optimization problem. This *one-stage* approach was first proposed by Lanckriet et al. (2004) and has since received significant attention (Rakotomamonjy et al., 2007; Sonnenburg et al., 2006; Cortes et al., 2010a; Kloft et al., 2011; Bach, 2008; Zien & Ong, 2007; Cortes et al., 2009a; Sindhwani & Lozano, 2011). The basic optimization problem in one stage MKL is given as

$$\min_{\boldsymbol{\mu} \in \mathcal{A}} \max_{\boldsymbol{\alpha}} \boldsymbol{\alpha}' \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}' \mathbf{Y}' \mathbf{K}'_{\boldsymbol{\mu}} \mathbf{Y} \boldsymbol{\alpha}, \quad \text{s.t. } 0 \leq \boldsymbol{\alpha} \leq C, \ \boldsymbol{\alpha}' \mathbf{y} = 0, \tag{2.6}$$

where $\mathbf{y}$ is a length n vector containing the class labels of training examples, $\mathbf{Y} = \text{diag}(\mathbf{y})$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]$, $\mathbf{K}_{\boldsymbol{\mu}}$ is the training kernel matrix parameterized by $\boldsymbol{\mu}$ and $\mathbf{1}$ is a vector of appropriate length containing all ones. The kernel quality in the optimization problem of Eq. 2.6 is measured by the SVM objective (as given in Eq. 2.5). If $\mathcal{A}$ is a convex set and the map $\boldsymbol{\mu} \mapsto \mathbf{K}_{\boldsymbol{\mu}}$ is a concave function in $\boldsymbol{\mu}$, the problem in convex in $\boldsymbol{\mu}$ as point-wise maximum preserves convexity. Nonnegative linear combination assumes $\mathbf{K}_{\boldsymbol{\mu}} = \sum_{i=1}^{p} \mu_i \mathbf{K}_i,\ \boldsymbol{\mu} \geq 0$ where p is the number of base kernels (Lanckriet et al., 2004). There has been considerable amount of work investigating other possibilities for set $\mathcal{A}$, e.g., $\boldsymbol{\mu}$ lying on the simplex (Sonnenburg et al., 2006; Rakotomamonjy et al., 2007; Zien & Ong, 2007), $\|\boldsymbol{\mu}\|_q$ norm penalty on the combination weights (Kloft et al., 2011; Orabona & Jie, 2011) for any $q \geq 1$, etc. A significant effort has also been invested in coming up with more efficient optimization methods for various MKL formulations having basic skeleton of Eq. 2.6 (Sonnenburg et al., 2006; Rakotomamonjy et al., 2007).

2.3.2 Two stage MKL

The second line of work in kernel learning follows a two-stage approach: first learn a "good" combination of base kernels using the training data, then use the learned kernel with a standard kernel method such as SVM to obtain a classifier. This approach has been initially proposed in Cristianini et al. (2001) and Kandola et al. (2002), and recently revisited by Cortes et al. (2010b). The two-stage kernel leaning approaches so far have been based on the notion of *target alignment*. Intuitively, target alignment is a measure of similarity (agreement) between a kernel and the *target kernel*, which is derived from the training labels and represents the optimal kernel for the training sample. The target alignment approach proposed in Cortes et al. (2010b) solves the following optimization problem to get the combination weights

$$\max_{\boldsymbol{\mu} \in \mathcal{A}} \frac{\langle \mathbf{K}_{\boldsymbol{\mu}_c}, \mathbf{y}\mathbf{y}' \rangle}{\|\mathbf{K}_{\boldsymbol{\mu}_c}\|_F}, \tag{2.7}$$

where $\mathcal{A} = \{\boldsymbol{\mu} : \boldsymbol{\mu} \geq 0,\ \|\boldsymbol{\mu}\|_2 = 1\}$, $\mathbf{K}_{\boldsymbol{\mu}} = \sum_{i=1}^{p} \mu_i \mathbf{K}_i$ and $\mathbf{K}_{\boldsymbol{\mu}_c}$ is the centered version of $\mathbf{K}_{\boldsymbol{\mu}}$. The matrix $\mathbf{y}\mathbf{y}'$ is taken as the target kernel whose (ij)th entry is $+1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are from same class and -1 otherwise. This target kernel is optimal for the training samples since the prediction for a training sample $\mathbf{x}$ reduces to $\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) = \text{sign}(\mathbf{x}) \sum_{i=1}^{n} \alpha_i$ which is always correct irrespective of the $\boldsymbol{\alpha}$ as $\alpha_i \geq 0$. We want the learned kernel $\mathbf{K}_{\mu}$ to be as close as possible to the target kernel on the training samples, and due to high probability concentration bounds (probability of deviation of a random variable from its mean decaying fast enough) we can hope to maintain a high target alignment on the unseen test examples as well (in expectation) (Cortes et al., 2010b).

In spite of a large amount of work over the past ten years (or so) on the problem of MKL, it is still an active research area, mainly due to the fact that the MKL methods have not yet realized their full potential that the research community believes they possess. MKL has achieved significant empirical success in a few application areas like bio-informatics and computer vision but it is still far from the state where MKL can be used as a reliable black-box in a variety of applications. In fact, it has become a challenge to make MKL perform better than simple well-established baselines like averaging the base kernels and best kernel selection using cross-validation. For a detailed survey of the existing multiple kernel learning algorithms, the reader is referred to Gonen & Alpaydin (2011).

Part I

Spectral Embeddings from Multiple Similarity Graphs

Chapter 3

Co-trained Spectral Embedding

This chapter describes an approach to learn spectral embeddings from multiple similarity graphs. The proposed co-training based framework can be seen as a multi-view generalization of the spectral embedding proposed in Belkin & Niyogi (2003). The approach is presented in the context of spectral clustering (using a particular type of graph Laplacian), which essentially learns spectral embedding followed by k-means clustering. However, the proposed co-training based framework is more general and can be used for general dimensionality reduction tasks.

3.1 Co-training

Co-training was originally proposed for the problem of semi-supervised learning, where we have access to labeled as well as unlabeled data (Blum & Mitchell, 1998). It considers a setting in which each example can be partitioned into two distinct views, and makes three main assumptions for its success: (a) *Sufficiency:* Each view is sufficient for classification on its own, (b) *Compatibility:* the target functions in both views predict same labels for co-occurring features with high probability, and (c) *Conditional independence:* the views are conditionally independent given the class label. Some of these assumptions, including conditional independence, were significantly relaxed in Balcan et al. (2004).

The central idea of co-training algorithms is to limit the search for target hypothesis to the set of "compatible hypotheses" that predict same labels for co-occurring patterns in each view. Unlabeled data allows us to do this pruning of the hypothesis space. In the original co-training algorithm (Blum & Mitchell, 1998), two initial hypotheses h_1 and h_2 are trained in the individual views using the labeled data. Both hypotheses then label a certain number of unlabeled examples on which they are most confident. These examples are added to the labeled pool, and h_1, h_2 are retrained. This process is repeated for a pre-chosen number of iterations. The intuition behind co-training algorithm is that h_1 adds examples to the labeled set that are used for training h_2, and vice versa. This process should slowly drive h_1 and h_2 to agree with each other on labels.

Variants of the original co-training algorithm were also proposed later and evaluated on different datasets. We specifically mention the *co-EM* algorithm (Nigam & Ghani, 2000). It differs with the original co-training algorithm in a couple of places. Firstly, it is not incremental in nature, i.e., all of unlabeled data is labeled in each iteration for further use. Secondly, only the data labeled by h_2 is used to retrain h_1 (and vice versa), unlike the original co-training algorithm that uses data labeled by both h_1 and h_2 in retraining each of these. Nigam & Ghani (2000) observe that co-EM is a closer match to the theoretical argument of Blum & Mitchell (1998) than the original co-training algorithm.

3.2 Co-training for Spectral Clustering

In this section, the idea of co-training is applied to the problem of multi-view spectral clustering. There is no labeled data in unsupervised learning problems, so semi-supervised co-training cannot be applied directly. However, the motivation still remains the same as in semi-supervised problems: to limit our search to hypotheses (in our problem, clusterings) that agree with those in other views. Specifically, we want the relationship within a pair of points to be consistent across the views. If two points are assigned in same cluster in one view, it should be so in all the views. On the other hand, if two points belong to different clusters, it should be so in all the views. This is a reasonable approach to take in the light of compatibility assumption of co-training.

We know that the first k eigenvectors of a graph Laplacian with exactly k number of connected components are the component (or cluster) indicator vectors, i.e., each vector is associated to a cluster and has non-zero values only at positions that correspond to points in the cluster. In another words, these eigenvectors only contain discriminative information about the clusters, ignoring the within-cluster details. For a fully connected graph (one connected component), spectral clustering solves a relaxed version of the min-cut problem (normalized or unnormalized). The eigenvectors in this case are not the cluster indicator vectors, yet they still contain discriminative information which is used in spectral clustering. In the multi-view setting, we can make use of eigenvectors obtained from one view to "label" the points in other view, and vice versa. Our proposed multi-view algorithm aims to work along the lines of Figure 3.1.

1. Solve spectral clustering on individual graphs to get the discriminative eigenvectors in each view, say $\mathbf{U}_1$ and $\mathbf{U}_2$.
2. Cluster points using $\mathbf{U}_1$ and use this clustering to *modify* the graph structure in view 2.
3. Cluster points using $\mathbf{U}_2$ and use this clustering to *modify* the graph structure in view 1.
4. Go to Step 1 and repeat for a number of iterations.

Figure 3.1: General framework for co-training based clustering

Now, the question remains of how to *modify* the graph structure using clustering information from the other view. One naïve way could be to reduce the edge-weight of a pair in a graph if its points belong to different clusters according to the other view. Alternatively, we can amplify the edge-weight of a pair if the other view puts it in same cluster. A similar idea could be applied for the other graph. However, this would require us to cluster points at each step, which may not be computationally efficient. In addition, there is also a question of how to decide the amounts or factors by which to reduce the different edge weights.

Instead of completely solving the clustering at each iteration and then "labeling" the other graph, an indirect approach is taken that results in extra computational savings, and is also more elegant. For a similarity matrix $\mathbf{K}_{n\times n}$, we can consider each column $\mathbf{k}_i$ of it as an n-dimensional vector that indicates the similarities of ith point with all the points in the graph. The eigenvectors of the graph Laplacian are vectors in the n-dimensional space. Since we know that the first k eigenvectors have the discriminative information for clustering, we can project the similarity vectors along these *directions* to retain the information needed for clustering and *throw away* the within cluster details that might confuse us in clustering. We back-project to the original n-dimensional space to get back the modified graph. Since the projection matrix is orthogonal, the inverse projection can be done using its transpose. This process is equivalent to steps 2 and 3 in

Figure 3.1. Algorithm 1 gives a detailed description of the algorithm. A symmetrization step is performed on $\mathbf{S}_1$ and $\mathbf{S}_2$ in the algorithm since the projection of similarity matrix $\mathbf{K}$ on the eigenvectors does not yield a symmetric matrix. Symmetrization operator on a matrix $\mathbf{S}$ is defined as $sym(\mathbf{S}) = (\mathbf{S} + \mathbf{S}^T)/2$.

Algorithm 1 Co-trained Multi-view Spectral Clustering

Input:
Similarity matrix for both views: $\mathbf{K}_1$, $\mathbf{K}_2$
Output: Assignments to k clusters
Initialize: $\mathbf{L}_v = \mathbf{D}_v^{-1/2}\mathbf{K}_v\mathbf{D}_v^{-1/2}$ for $v = 1, 2$
$\mathbf{U}_v^0 = \arg\max \mathbf{U} \in \mathbb{R}^{n\times k}\, tr(\mathbf{U}^T\mathbf{L}_v\mathbf{U})$, s.t. $\mathbf{U}^T\mathbf{U} = I$ for $v = 1, 2$
for $i = 1$ to $iter$ **do**
1: $\mathbf{S}_1 = sym\left(\mathbf{U}_2^{i-1}\mathbf{U}_2^{i-1^T}\mathbf{K}_1\right)$
2: $\mathbf{S}_2 = sym\left(\mathbf{U}_1^{i-1}\mathbf{U}_1^{i-1^T}\mathbf{K}_2\right)$
3: Use $\mathbf{S}_1$ and $\mathbf{S}_2$ as the new graph similarities and compute the Laplacians. Solve for the largest k eigenvectors to obtain $\mathbf{U}_1^i$ and $\mathbf{U}_2^i$.
end for
4: Row-normalize $\mathbf{U}_1^i$ and $\mathbf{U}_2^i$.
5: Form matrix $\mathbf{V} = \mathbf{U}_v^i$, where v is believed to be the most informative view a priori. If there is no prior knowledge on the view informativeness, matrix $\mathbf{V}$ can also be set to be column-wise concatenation of the two $\mathbf{U}_v^i$s.
6: Assign example j to cluster c if the j-th row of $\mathbf{V}$ is assigned to cluster c by the k-means algorithm.

To further reinforce the idea of projection along the eigenvectors, let us consider a simple case where the first graph has exactly k components in it, i.e., the weights of across cluster edges are 0. As we know, the Laplacian of this graph would have the top k eigenvectors as the cluster indicator vectors (von Luxburg, 2007). Let us assume that the second view has a fully connected graph with similarity matrix as follows:

$$\mathbf{K}_2 = \begin{pmatrix} 0 & b & c & d \\ b & 0 & e & f \\ c & e & 0 & g \\ d & f & g & 0 \end{pmatrix} \tag{3.1}$$

The self-similarities of all points are assumed to be same and equal to 0, since they do not affect the min-cut solution. Suppose the second graph gives $\mathbf{u}_1^1 = \frac{1}{\sqrt{3}}(1\,1\,1\,0)^T$ and $\mathbf{u}_1^2 = (0\,0\,0\,1)^T$ as the top two eigenvectors. This implies that first 3 points are in one cluster and the fourth point is in the second cluster. Let $\mathbf{U}_1 = \left(\mathbf{u}_1^1\ \mathbf{u}_1^2\right)$. The projection of $\mathbf{K}_2$ onto the subspace spanned by $\mathbf{U}_1$ and the subsequent symmetrization yields the following modified graph in view 2:

$$\tfrac{1}{3}\begin{pmatrix} b{+}c & b{+}(c{+}e)/2 & c{+}(b{+}e)/2 & 2d{+}(f{+}g)/2 \\ b{+}(c{+}e)/2 & b{+}e & e{+}(b{+}c)/2 & 2f{+}(d{+}g)/2 \\ c{+}(b{+}e)/2 & e{+}(b{+}c)/2 & c{+}e & 2g{+}(d{+}f)/2 \\ 2d{+}(f{+}g)/2 & 2f{+}(d{+}g)/2 & 2g{+}(d{+}f)/2 & 0 \end{pmatrix}$$

Let us pay attention to the shaded sub-matrix in the above matrix. The new weight of edge (i, j) is obtained by averaging out the edges within the cluster. The across cluster edges are also averaged

out in the new graph. This implies that the projection in the subspace of discriminative eigenvectors makes edges within a cluster close to each other, throwing away the intra-cluster information that is irrelevant for clustering. As the number of iterations increase, the edges within a cluster diffuse to one another. The across cluster edges also diffuse to one another. All points within a cluster are treated in a similar way, and different from points in other clusters. In other words, this process "glues" all the points in a cluster, and they tend to appear together in the subsequent spectral clustering solution.

It is possible to extend the proposed co-training framework for more than two views. We can take the similarity matrix $\mathbf{K}_v$ of a view, and project it onto the union of subspaces spanned by top k discriminative eigenvectors of the other views. More formally, steps 1 and 2 in Algorithm 1 are replaced by $\mathbf{S}_v = sym\left(\left(\sum_{i \neq v} \mathbf{U}_i \mathbf{U}_i^T\right) \mathbf{K}_v\right)$ for all the views.

3.2.1 Computational Efficiency

The projection of the similarity matrix $\mathbf{K}$ using the projection matrix $\mathbf{U}\mathbf{U}^T$ gives a matrix that has a rank of k. After symmetrization, each of the new similarity matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ has a maximum rank of $2k$. Hence, the normalized Laplacian $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$ also has a maximum rank of $2k$. This can be of advantage in large scale problems, since there exist efficient randomized algorithms for doing SVD if the original matrix is low rank and a good upper bound on the rank is known in advance (Liberty et al., 2007). The matrix $\tilde{\mathbf{S}} = \mathbf{U}\mathbf{U}^T K$ is a diagonalizable matrix and has all real non-negative eigenvalues (refer to Theorem 7.6.3 in (Horn & Johnson, 1990)). However, it is not necessary for $sym(\mathbf{S}) = (\tilde{\mathbf{S}} + \tilde{\mathbf{S}}^T)/2$ to have non-negative eigenvalues. The individual entries of $sym(\mathbf{S})$ can also be negative, and so the corresponding Laplacian can be non-positive definite. In the experiments, a rank-1 matrix is added to $sym(\mathbf{S})$ that has all its entries equal to the minimum negative entry of $sym(\mathbf{S})$. This makes sure that the corresponding Laplacian is positive semidefinite at each iteration.

3.3 When will co-training help in spectral clustering?

Blum & Mitchell (1998) state the assumptions under which co-training will succeed for classification (also mentioned in Sec. 3.1). It was shown later by Balcan et al. (2004) that these assumptions can be substantially weakened with stronger assumptions on the learning algorithm for co-training to succeed. In this section, I will construct some example scenarios to better understand when co-training can be expected to help in spectral clustering. I also construct some negative examples for which co-training does not help.

3.3.1 Correlated Features

Here I simulate the scenario when we have correlated features across views given the cluster label. Suppose there are two views and two clusters in each view. Data points in each view lie in two dimensions. To generate a multiview sample, first a cluster label is sampled from a Bernoulli distribution. Then, depending on the cluster label, a multiview sample is generated according to the corresponding Gaussian distribution. The mean vectors and covariance matrices for the two clusters are given as follows, where μ_i and Σ_i are the mean and covariance for cluster i, with the first two dimensions corresponding to the first view and the last two dimensions corresponding to

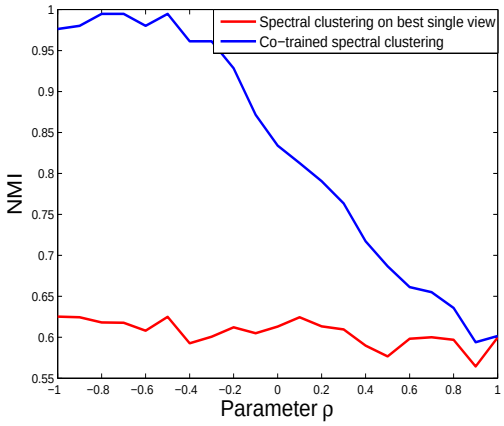

Figure 3.2: NMI scores for best single view and for multiview co-trained spectral clustering vs the feature correlation ρ

the second view. After generating a multiview sample x, it is split into two views as $x^{(1)} = x(1:2)$ and $x^{(2)} = x(3:4)$.

$$\mu_1 = (1\ \ 1\ \ 1\ \ 1)\,, \qquad \mu_2 = (3\ \ 3\ \ 3\ \ 3)$$

$$\Sigma_1 = \begin{pmatrix} 1 & 0 & \rho & 0 \\ 0 & 1 & 0 & \rho \\ \rho & 0 & 1 & 0 \\ 0 & \rho & 0 & 1 \end{pmatrix}, \qquad \Sigma_2 = \begin{pmatrix} 1 & 0 & \rho & 0 \\ 0 & 1 & 0 & \rho \\ \rho & 0 & 1 & 0 \\ 0 & \rho & 0 & 1 \end{pmatrix}$$

The parameter ρ controls the correlation between the features of two views given the cluster label. I vary the parameter ρ from -1 to 1 in the steps of 0.1. For each value of ρ, 1000 multiview data points are generated as discussed above and split into two views. Gaussian kernel is used for computing the graph similarities in each view. The standard deviation of the kernel is taken to be the median of the pair-wise Euclidean distances between the data points. The plot of normalized mutual information (NMI) versus the parameter ρ is shown in Fig. 3.2. A higher value of NMI indicates better clustering accuracy. Please see the next section for details regarding NMI and other clustering measures. The red and blue plots show the NMI obtained with best single view and with co-trained spectral clustering algorithm, respectively. **The improvement obtained with co-training decreases with the increase in the parameter ρ.** This is not unexpected: a negative correlation between features in the two views in this case ensures that if a sample is more confusable with the other cluster in one view, it will be less confusable with the other cluster in the second view. Thus each view provides complementary information that helps improve the clustering. Totally uncorrelated features ($\rho = 0$) also help co-training improve beyond the best single view. On the other hand, when ρ is high, both views provide similar information and we do not gain much by the use of co-training.

3.3.2 Complementary corruption in the similarity graphs

Let us start with two identical similarity matrices of block diagonal structure, one for each view.

$$K^{(1)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}, \qquad K^{(2)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}$$

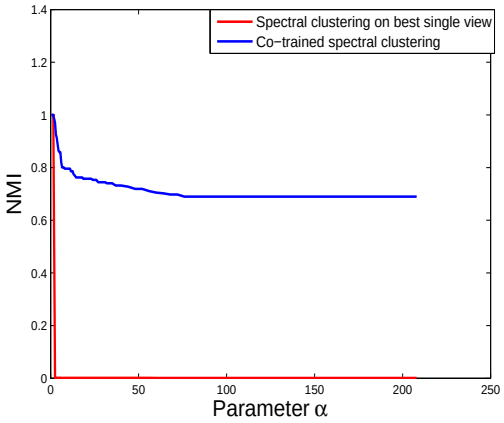

Figure 3.3: NMI scores for best single view and for multiview co-trained spectral clustering vs the noise level α

Here both $K^{(1)}$ and $K^{(2)}$ are of size $n \times n$, the matrix $\mathbf{1}$ is a matrix of all 1's of size $\frac{n}{2} \times \frac{n}{2}$ and matrix $\mathbf{0}$ is a matrix of all 0's. I assume a ground truth clustering that first $\frac{n}{2}$ points belong to one cluster and the last $\frac{n}{2}$ points belong to the second cluster. The two similarity matrices are identical so we will not gain anything by using two views in clustering. Now, let us inject noise in the off-diagonal blocks of both similarity matrices. I randomly select half of the entries in the off-diagonal blocks of $K^{(1)}$ (maintaining the symmetry of the matrix) and put a similarity value of α in these positions. I also put similarity values of α in the off-diagonal blocks of $K^{(2)}$ but in the complementary positions (where $K^{(1)}$ off-diagonal blocks have zeros). Thus, for every position (i, j) in the off-diagonal blocks, one of similarity matrices has α while the other has 0. The number of points n is taken to be 1000 and the parameter α is varied from 0.5 to 200. The plot of NMI against variation in α is shown in Fig. 3.3. The NMI obtained with single view spectral clustering is 1 for $\alpha \leq 1.5$ and suddenly decreases to the order of 10^{-3} at $\alpha = 2$ observing a phase transition. **The co-trained spectral clustering is able to improve the NMI making use of the complementary information in the similarity matrices of the two views.** Note that the simple baseline of "similarity addition" (adding the two similarity matrices and doing spectral clustering on the resultant matrix) does not work for this case and actually gives worse NMI than single view spectral clustering. However, another simple baseline of "similarity product" (taking the Hadamard product of the two similarity matrices) will work well for this case due to the particular nature of corruption.

3.3.3 Low quality or contradictory individual views

Two of the assumptions in semi-supervised co-training are that each view should be sufficient on its own to give a reasonably good classification performance (sufficiency) and the target functions in both views predict same labels for co-occurring features with high probability (compatibility). Here I test these assumptions for co-trained spectral clustering. To this end, I simulate a scenario where one of the similarity matrices does not have any useful information for clustering or may even indicate a different clustering of data. The similarity matrices in the two views are

given as follows.

$$K^{(1)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}, \qquad K^{(2)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}$$

I assume a ground truth clustering that first $\frac{n}{2}$ points belong to one cluster and the last $\frac{n}{2}$ points belong to the second cluster. The matrix $K^{(2)}$ has equal similarity for all pairs of points and does not have any useful information for clustering. For this case, the first view gives a perfect NMI of 1 and the second view gives extremely low NMI of the order of 10^{-4}. The co-trained spectral clustering gives an NMI of 0.35. This example illustrates that co-training will not work well in the presence of a "bad" view as it can influence the subsequent iterations of the co-trained spectral clustering algorithm.

As a second example, consider the following similarity matrices for the two views that indicate contradictory clusterings.

$$K^{(1)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}, \qquad K^{(2)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{4}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{4}} \\ \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} \end{pmatrix}$$

These similarity matrices indicate significantly different clustering assignments. In this case, the co-trained spectral clustering algorithm oscillates heavily across its iterations and does not help. Of course, no multiview algorithm can be expected to work in this case since the two views admit significantly different clusterings and no sharing is possible.

3.4 Experiments

In this section, the co-trained multi-view spectral clustering approach is compared with the following baselines.

- **Single View:** Using the most informative view, i.e., one that achieves the best spectral clustering performance using a single view of the data.
- **Feature Concatenation:** Concatenating the features of each view, and then running standard spectral clustering using the graph Laplacian derived from the joint view representation of the data.
- **Kernel Addition:** Combining different kernels by adding them, and then running standard spectral clustering on the corresponding Laplacian. As suggested in earlier findings (Cortes et al., 2009b), even this seemingly simple approach often leads to near optimal results as compared to more sophisticated approaches for classification. It can be noted that kernel addition reduces to feature concatenation for the special case of linear kernel. In general, kernel addition is same as concatenation of features in the Reproducing Kernel Hilbert Space.
- **Kernel Product (element-wise):** Multiplying the corresponding entries of kernels and applying standard spectral clustering on the resultant Laplacian. For the special case of Gaussian kernel, element-wise kernel product would be same as simple feature concatenation if both kernels use same width parameter σ. However, in the experiments, different width parameters are used for different views so the performances of kernel product may not be directly comparable to feature concatenation.

Method	F-score	Precision	Recall	Entropy	NMI	Adj-RI
Best Single View	0.971	0.975	0.968	0.097	0.898	0.942
Feature Concat	0.980	0.983	0.977	0.068	0.928	0.960
Kernel Addition	0.996	0.996	0.996	0.020	0.973	0.992
Kernel Product	0.990	0.988	0.991	0.041	0.959	0.980
CCA	0.984	0.984	0.984	0.067	0.932	0.968
Min-Disagreement	0.984	0.986	0.983	0.062	0.936	0.968
Co-trained spectral(2)	0.996	0.995	0.996	0.019	0.981	0.992
Co-trained spectral(3)	0.998	0.998	0.997	0.010	0.989	0.996

Table 3.1: Clustering performance on **synthetic data**. Number (2) or (3) indicates the number of views used in the approach. Std. deviations of all performance metrics are zero for this synthetic data.

- **CCA based Feature Extraction:** Applying CCA for feature fusion from multiple views of the data (Blaschko & Lampert, 2008), and then running spectral clustering using these extracted features. I apply both standard CCA and kernel CCA for feature extraction and report the clustering results for whichever method gives the best performance on test data.
- **Minimizing-Disagreement Spectral Clustering:** Last baseline is the *minimizing-disagreement* approach to spectral clustering (de Sa, 2005), and is perhaps most closely related to the propsoed co-training based approach to spectral clustering. This algorithm is discussed more in Sec. 3.5.

I report experimental results on one synthetic and three real-world datasets.

- **Synthetic data:** The synthetic data consists of three views and is generated as follows. I first choose the cluster c_i each sample belongs to, and then generate each of the views $x_i^{(1)}$, $x_i^{(2)}$ and $x_i^{(3)}$ from a two-component Gaussian mixture model. These views are combined to form the sample $(x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, c_i)$. Total 1000 points are sampled from each view. The cluster means in view 1 are $\mu_1^{(1)} = (1\ \ 1)$, $\mu_2^{(1)} = (3\ \ 4)$; in view 2 are $\mu_1^{(2)} = (1\ \ 2)$, $\mu_2^{(2)} = (2\ \ 2)$; and in view 3 are $\mu_1^{(3)} = (1\ \ 1)$, $\mu_2^{(3)} = (3\ \ 3)$. The covariances for the three views are given below. The notation $\Sigma_c^{(v)}$ denotes the parameter for cth cluster in vth view.

$$\Sigma_1^{(1)} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}, \quad \Sigma_2^{(1)} = \begin{pmatrix} 0.3 & 0.2 \\ 0.2 & 0.6 \end{pmatrix}$$
$$\Sigma_1^{(2)} = \begin{pmatrix} 1 & -0.2 \\ -0.2 & 1 \end{pmatrix}, \quad \Sigma_2^{(2)} = \begin{pmatrix} 0.6 & 0.1 \\ 0.1 & 0.5 \end{pmatrix}$$
$$\Sigma_1^{(3)} = \begin{pmatrix} 1.2 & 0.2 \\ 0.2 & 1 \end{pmatrix}, \quad \Sigma_2^{(3)} = \begin{pmatrix} 1 & 0.4 \\ 0.4 & 0.7 \end{pmatrix}$$

- **Reuters Multilingual data:** The test collection contains feature characteristics of documents originally written in five different languages (English, French, German, Spanish and Italian), and their translations, over a common set of 6 categories (Amini et al., 2009). I use documents originally in English as the first view, and their French and German translations as the second and third views. I randomly sample 1200 documents from this collection in a balanced manner, with each of the 6 clusters having 200 documents. The documents are in bag-of-words representation which implies that the features are extremely sparse and

Method	F-score	Precision	Recall	Entropy	NMI	Adj-RI
Best Single View	0.342(0.010)	0.296(0.015)	0.407(0.025)	1.878(0.052)	0.287(0.019)	0.186(0.014)
Feature Concat	0.368(0.012)	0.330(0.016)	0.416(0.017)	1.841(0.057)	0.298(0.020)	0.225(0.017)
Kernel Addition	0.386(0.012)	0.358(0.017)	0.420(0.023)	1.770(0.058)	0.323(0.021)	0.252(0.016)
Kernel Product	0.258(0.003)	0.198(0.011)	0.381(0.058)	2.306(0.034)	0.123(0.010)	0.052(0.014)
CCA	0.262(0.007)	0.222(0.005)	0.322(0.034)	2.232(0.009)	0.147(0.003)	0.082(0.003)
Min-Disagreement	0.381(0.014)	0.341(0.004)	0.435(0.035)	1.736(0.052)	0.342(0.024)	0.240(0.012)
Co-trained spectral(2)	0.401(0.009)	0.363(0.007)	0.450(0.030)	1.651(0.024)	0.373(0.012)	0.267(0.007)
Co-trained spectral(3)	0.412(0.001)	0.369(0.001)	0.467(0.003)	1.616(0.017)	0.388(0.007)	0.279(0.001)

Table 3.2: Clustering performance on **Reuters multilingual data**. The languages used are English, French, and German. Number (2) or (3) indicates the number of views used in the approach. Numbers in parentheses are the std. deviations.

Method	F-score	Precision	Recall	Entropy	NMI	Adj-RI
Best Single View	0.577(0.015)	0.569(0.020)	0.586(0.012)	1.198(0.029)	0.641(0.008)	0.530(0.017)
Feature Concat	0.536(0.027)	0.514(0.026)	0.561(0.032)	1.283(0.050)	0.619(0.015)	0.480(0.026)
Kernel Addition	0.707(0.052)	0.688(0.065)	0.727(0.037)	0.862(0.110)	0.744(0.030)	0.673(0.059)
Kernel Product	0.719(0.049)	0.698(0.064)	0.742(0.032)	0.832(0.102)	0.754(0.026)	0.687(0.055)
CCA	0.638(0.027)	0.616(0.037)	0.662(0.020)	1.073(0.071)	0.682(0.019)	0.596(0.031)
Min-Disagreement	0.693(0.047)	0.663(0.066)	0.729(0.026)	0.870(0.096)	0.745(0.024)	0.658(0.053)
Co-trained spectral	0.726(0.048)	0.709(0.058)	0.745(0.039)	0.793(0.109)	0.765(0.031)	0.695(0.054)

Table 3.3: Clustering performance on **Handwritten digits data**. Numbers in parentheses are the std. deviations.

high-dimensional. The standard similarity measures (like Gaussian kernel) in very high dimensions are often unreliable. Since spectral clustering essentially works with similarities of the data, we first project the data using Latent Semantic Analysis (LSA) (Hofmann, 1999) to a 100-dimensional space and compute similarities in this lower dimensional space. This is akin to a computing topic based similarity of documents (Blei et al., 2003).

- **UCI Handwritten digits data:** The second real-world dataset is taken from the handwritten digits (0-9) data from the UCI repository. The dataset consists of 2000 examples, with view-1 being the 76 Fourier coefficients, and view-2 being the 216 profile correlations of each example image.
- **BBC and BBCSPORTS data:** These datasets consist of news articles from the BBC (Greene & Cunningham, 2005). BBC data contains 2225 complete news articles corresponding to stories in five topical areas (business, entertainment, politics, sport, tech). BBCSPORTS data consists of 737 sports news articles in five classes (athletics, cricket, football, rugby, tennis). These are synthetic multi-view datasets, wherein each document is segmented and segments are randomly assigned to the two views (Greene & Cunningham, 2009).

The approaches are compared on a number of evaluation measures. Here, I report precision, recall, F-score, normalized mututal information (NMI), average entropy, and adjusted rand index (Manning et al., 2008; Hubert & Arabie, 1985). For all these measures, the higher value indicates better clustering quality, except for average cluster entropy, for which lower value signifies better clustering quality. Each evaluation measure penalizes or favors different properties in the clustering, hence I report results on these diverse measures to do a comprehensive evaluation.

Method	F-score	Precision	Recall	Entropy	NMI	Adj-RI
Best Single View	0.546(0.001)	0.522(0.001)	0.572(0.001)	1.221(0.003)	0.478(0.001)	0.424(0.001)
Feature Concat	0.559(0.019)	0.526(0.016)	0.598(0.022)	1.152(0.027)	0.512(0.013)	0.439(0.024)
Kernel Addition	0.558(0.013)	0.525(0.013)	0.595(0.013)	1.166(0.016)	0.506(0.008)	0.437(0.017)
Kernel Product	0.572(0.033)	0.536(0.028)	0.614(0.039)	1.132(0.053)	0.522(0.025)	0.455(0.042)
CCA	0.220(0.001)	0.193(0.001)	0.257(0.002)	1.861(0.003)	0.214(0.001)	0.178(0.001)
Min-Disagreement	0.854(0.047)	0.849(0.065)	0.860(0.026)	0.479(0.093)	0.794(0.033)	0.816(0.062)
Co-trained spectral	0.898(0.000)	0.894(0.000)	0.902(0.000)	0.369(0.000)	0.841(0.000)	0.873(0.000)

Table 3.4: Clustering performance on **BBC data**. Numbers in parentheses are the std. deviations.

Method	F-score	Precision	Recall	Entropy	NMI	Adj-RI
Best Single View	0.387(0.015)	0.405(0.021)	0.370(0.015)	1.565(0.066)	0.286(0.028)	0.210(0.022)
Feature Concat	0.609(0.040)	0.636(0.019)	0.585(0.059)	0.919(0.009)	0.575(0.004)	0.497(0.047)
Kernel Addition	0.604(0.038)	0.634(0.020)	0.578(0.054)	0.898(0.014)	0.584(0.004)	0.491(0.045)
Kernel Product	0.603(0.036)	0.635(0.018)	0.575(0.053)	0.910(0.011)	0.578(0.003)	0.490(0.043)
CCA	0.173(0.008)	0.187(0.010)	0.161(0.006)	1.89(0.074)	0.115(0.011)	0.089(0.005)
Min-Disagreement	0.718(0.082)	0.751(0.051)	0.690(0.109)	0.646(0.080)	0.697(0.045)	0.638(0.102)
Co-trained spectral	0.850(0.078)	0.866(0.057)	0.836(0.095)	0.392(0.091)	0.817(0.047)	0.807(0.099)

Table 3.5: Clustering performance on **BBCSPORTS data**. Numbers in parentheses are the std. deviations.

Gaussian kernel is used for computing the graph similarities in all the experiments. The standard deviation of the kernel is taken equal to the median of the pair-wise Euclidean distances between the data points, except for the BBC data for which it gives extremely low performance. I use a kernel std. dev. of 100 for both BBC datasets. In all the result tables, the numbers in the parentheses are the standard deviations of the performance measures obtained with 20 different runs of k-means with random initializations.

The results for synthetic data are shown in Table 3.1. As it can be seen, the proposed approach outperforms all the baselines. Baselines are run using first using two views and then using all three views, and the best results are reported here. The closest performing approach is kernel addition. For synthetic data, order-2 polynomial kernel based kernel-CCA gives best performance among all CCA variants, while Gaussian kernel based kernel-CCA performs poorly. I do not report results for Gaussian kernel CCA here. All the baselines outperform the single view case for the synthetic data.

Table 3.2 shows the document clustering results on Reuters multilingual data with English, French and German documents as the three views. On this dataset too, the proposed approach outperforms all the baselines by a significant margin. The next best performance is attained by minimum-disagreement spectral clustering (de Sa, 2005) approach. It should be noted that CCA and element-wise kernel product performances are worse than that of single view.

Table 3.3 shows the results on UCI Handwritten digits dataset. On this dataset, quite a few approaches including kernel addition, element-wise kernel multiplication, and minimum-disagreement are close to the co-training based spectral clustering approach. The proposed approach still manages to perform marginally better than the best of these on all evaluation metrics. It can be noted that feature concatenation performs worse than single view.

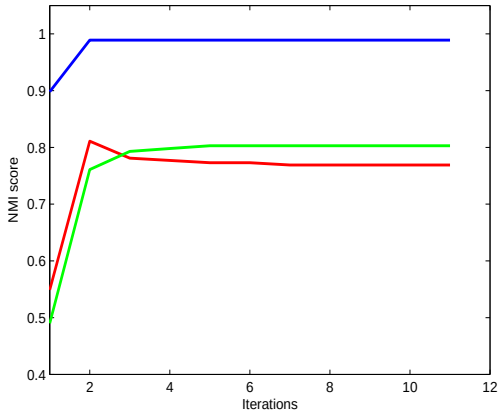

Figure 3.4: NMI scores in different views vs number of iterations of co-trained spectral clustering for **Synthetic data**

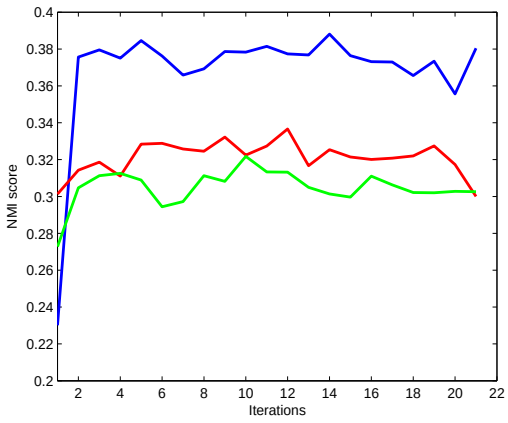

Figure 3.5: NMI scores in different views vs number of iterations of co-trained spectral clustering for **Reuters multilingual data**

Finally, the results on BBC and BBCSPORTS data are shown in Tables 3.4 and 3.5. All baselines perform better than single view. Our proposed approach outperforms the closest performing baseline, which is minimum-disagreement approach, by a significant margin. CCA again performs worse than single view as was the case for Reuters multilingual data.

I also show the variation in NMI score as the number of iterations increase in Figures 3.4 and 3.5. For the synthetic data, algorithm converges after four iterations for all the three views and remains constant after that. For Reuters data, the major improvement in performance for all views is obtained after the first iteration. It keeps varying around that value in the subsequent iterations. In general, it was observed that the algorithm does not converge, as is also the case with semi-supervised co-training algorithm, which is also not guaranteed to converge. In all the experiments, it was observed that the biggest increment in performance is obtained in the first iteration. I stop after a fixed number of iterations in the experiments. However, it is possible to apply some heuristic clustering performance measures (e.g. cluster compactness measure) to decide the stopping criterion.

3.5 Related Work

A number of clustering algorithms have been proposed in the past to learn with multiple views of the data. Some of them first extract a set of shared features from the multiple views and then apply any off-the-shelf clustering algorithm such as k-means on these features. The Canonical Correlation Analysis (CCA) (Chaudhuri et al., 2009; Blaschko & Lampert, 2008) based approach is an example of this. Alternatively, some other approaches exploit the multiple views of the data as part of the clustering algorithm itself. For example, (Bickel & Scheffer, 2004) proposed a Co-EM based framework for multi-view clustering in mixture models. Co-EM approach computes expected values of hidden variables in one view and uses these in the M-step for other view, and vice versa. This process is repeated until a suitable stopping criteria is met. The algorithm often does not converge.

Multi-view clustering algorithms have also been proposed in the framework of spectral clustering (Zhou & Burges, 2007; de Sa, 2005). In (Zhou & Burges, 2007), the authors obtain a graph cut which is good on average over the multiple graphs but may not be the best for a single graph. They give a random walk based formulation for the problem. (de Sa, 2005) approaches the problem of two-view clustering by constructing a bipartite graph from nodes of both views. Edges of the bipartite graph connect nodes from one view to those in the other view. Subsequently, they solve standard spectral clustering problem on this bipartite graph. In (Tang et al., 2009), the information from multiple graphs are fused using Linked Matrix Factorization.

Our approach is in contrast with several other existing works on multiple kernel learning (Cortes et al., 2009b; Zhao et al., 2009) that try to learn an optimal kernel matrix by a linear or non-linear combination of base kernel matrices. It is possible to apply these approaches to multi-view spectral clustering by learning a new similarity matrix (or kernel matrix) that combines information from multiple kernels, and then solving the relaxed min-cut problem on this new graph. In contrast, our approach solves the relaxed min-cut problem (using spectral clustering) on individual graphs and enforces these cuts (or clusters) to agree with each other. The multiple kernel learning based approach to multi-view spectral clustering can be considered to have a generative flavor in the sense that it focuses on the generation of a new graph, while our co-regularization based approach tries to explicitly minimize the disagreement between clusterings in different given graphs.

Consensus clustering approaches can also be applied to the problem of multi-view clustering (Strehl & Ghosh, 2002). These approaches do not generally work with original features. Instead, they take different clusterings of a dataset coming from different sources as input and reconcile them to find a final clustering.

3.6 Discussion

This chapter presented a multi-view spectral clustering approach using the idea of co-training, which has been widely used in semi-supervised learning problems. The general framework of the proposed algorithm is to learn the clustering in one view and use it to "label" the data in other view so as to modify the graph structure (similarity matrix). The modification to the graph is dictated by the discriminative eigenvectors and is achieved by projection along these directions.

The key assumption that the true underlying clustering is same for all views is safe in most scenarios, and is necessary for multi-view algorithms to succeed. This assumption can potentially be violated in situations where the data assumes more than one natural clustering, and different

clusterings become prominent in different views. However, in this work, I am not concerned with the problem of multiple clusterings so compatibility of clustering across views is safe to assume.

It is possible to extend the proposed framework to the case where some of the views have missing data. For missing data points, the corresponding entries in the similarity matrices would be unavailable. We can estimate these missing similarities by the corresponding similarities in other views. One possible approach to estimate the missing entry could be to simply average the similarities from views in which the data point is available. Proper normalization of similarities (possibly by Frobenius norm of the whole matrix) might be needed before averaging to make them comparable.

Chapter 4

Co-regularized Spectral Embedding

The previous chapter presented a co-training based formulation for learning spectral embeddings from multiple graphs using an assumption that object groupings are same in all the views. This chapter describes an approach to learn spectral embeddings from multiple similarity graphs using co-regularization ideas from semi-supervised learning. The proposed co-regularization based framework can also be seen as a multi-view generalization of the spectral embedding proposed in Belkin & Niyogi (2003). The approach is presented in the context of spectral clustering (using a particular type of graph Laplacian), however the proposed framework is more general and can be used for general dimensionality reduction tasks.

4.1 Co-regularized Spectral Clustering

Let $\mathbf{X}^{(v)} = \{\mathbf{x}_1^{(v)}, \mathbf{x}_2^{(v)}, \ldots, \mathbf{x}_n^{(v)}\}$ denote the examples in view v and $\mathbf{K}^{(v)}$ denote the similarity matrix of $\mathbf{X}^{(v)}$ in this view. I use the normalized graph Laplacians for both views: $\mathcal{L}^{(v)} = \mathbf{D}^{(v)^{-1/2}}\mathbf{K}^{(v)}\mathbf{D}^{(v)^{-1/2}}$ following Ng et al. (2002), however other variants (von Luxburg, 2007) can also be used without any change in the basic formulation. The *single view* spectral clustering algorithm of (Ng et al., 2002) solves the following optimization problem for the normalized graph Laplacian $\mathbf{L}^{(v)}$:

$$\max_{\mathbf{U}^{(v)} \in \mathbb{R}^{n \times k}} tr\left(\mathbf{U}^{(v)^T}\mathcal{L}^{(v)}\mathbf{U}^{(v)}\right), \quad \text{s.t.} \quad \mathbf{U}^{(v)^T}\mathbf{U}^{(v)} = I \tag{4.1}$$

where tr denotes the matrix trace. The rows of matrix $\mathbf{U}^{(v)}$ are the embeddings of the data points that can be given to the k-means algorithm to obtain cluster memberships. For a detailed introduction to both theoretical and practical aspects of spectral clustering, the reader is referred to (von Luxburg, 2007). The proposed multi-view spectral clustering framework builds on the standard spectral clustering with a single view, by appealing to the co-regularization framework typically used in the semi-supervised learning literature (Blum & Mitchell, 1998; Sindhwani et al., 2005).

Co-regularization in semi-supervised learning essentially works by making the hypotheses learned from different views of the data agree with each other on unlabeled data (Sindhwani et al., 2005). The framework employs two main assumptions for its success: (a) the true target functions in each view should agree on the labels for the unlabeled data *(compatibility)*, and (b) the views are independent given the class label *(conditional independence)*. The *compatibility* assumption allows us to shrink the space of possible target hypotheses by searching only over the compatible functions. The *independence* assumption makes it unlikely for compatible classifiers to agree on

wrong labels. In the case of clustering, this would mean that a data point in both views would be assigned to the correct cluster with high probability.

Here, two co-regularization based approaches are presented that make the clustering hypotheses on different graphs (i.e., views) agree with each other. The effectiveness of spectral clustering hinges crucially on the construction of the graph Laplacian and the resulting eigenvectors that reflect the cluster structure in the data. Therefore, an objective function is constructed that consists of the graph Laplacians from all the views of the data and regularized on the eigenvectors of the Laplacians such that the cluster structures resulting from each Laplacian look consistent across all the views.

The first co-regularization scheme (Section 4.1.1) enforces that the eigenvectors $\mathbf{U}^{(v)}$ and $\mathbf{U}^{(w)}$of a view pair (v, w) should have high pairwise similarity (using a pair-wise co-regularization criteria we will define in Section 4.1.1). The second co-regularization scheme (Section 4.1.3) enforces the view-specific eigenvectors to look similar by regularizing them towards a common *consensus* (centroid based co-regularization). The idea is different from previously proposed consensus clustering approaches (Strehl & Ghosh, 2002) that commit to individual clusterings in the first step and then combine them to a consensus in the second step. We optimize for individual clusterings as well as the consensus using a joint cost function.

4.1.1 Pairwise Co-regularization

In standard spectral clustering, the eigenvector matrix $\mathbf{U}^{(v)}$ is the data representation for subsequent k-means clustering step (with i'th row mapping to the original i'th sample). In the proposed objective function, the pairwise similarities of examples under the new representation (in terms of rows of $\mathbf{U}^{(\cdot)}$'s) are encouraged to be similar across all the views. This amounts to enforcing the spectral clustering hypotheses (which are based on the $\mathbf{U}^{(\cdot)}$'s) to be similar across all the views.

Let us work with two-view case for the ease of exposition. This will later be extended to more than two views. Consider the following cost function as a measure of disagreement between embeddings of two views:

$$D(\mathbf{U}^{(v)}, \mathbf{U}^{(w)}) = \left\| \frac{\mathbf{K}_{\mathbf{U}^{(v)}}}{||\mathbf{K}_{\mathbf{U}^{(v)}}||_F^2} - \frac{\mathbf{K}_{\mathbf{U}^{(w)}}}{||\mathbf{K}_{\mathbf{U}^{(w)}}||_F^2} \right\|_F^2. \tag{4.2}$$

$\mathbf{K}_{\mathbf{U}^{(v)}}$ is the similarity matrix for $\mathbf{U}^{(v)}$, and $||\cdot||_F$ denotes the Frobenius norm of the matrix. The similarity matrices are normalized by their Frobenius norms to make them comparable across views. Linear kernel, i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ is chosen as the similarity measure in Equation 4.2. This implies that we have $\mathbf{K}_{\mathbf{U}^{(v)}} = \mathbf{U}^{(v)} {\mathbf{U}^{(v)}}^T$. The reason for choosing linear kernel to measure similarity of $\mathbf{U}^{(\cdot)}$ is twofold. First, the similarity measure (or kernel) used in the Laplacian for spectral clustering has already taken care of the non-linearities present in the data (if any), and the embedding $\mathbf{U}^{(\cdot)}$ being real-valued cluster indicators, can be considered to obey linear similarities. Secondly, we get a nice optimization problem by using linear kernel for $\mathbf{U}^{(\cdot)}$. It can be noticed that $||\mathbf{K}_{\mathbf{U}^{(v)}}||_F^2 = k$, where k is the number of clusters. Substituting this in Equation 4.2 and ignoring the constant additive and scaling terms that depend on the number of clusters, we get

$$D(\mathbf{U}^{(v)}, \mathbf{U}^{(w)}) = -tr\left(\mathbf{U}^{(v)} {\mathbf{U}^{(v)}}^T \mathbf{U}^{(w)} {\mathbf{U}^{(w)}}^T\right)$$

We want to minimize the above disagreement between the embeddings of views v and w. Combining this with the spectral clustering objectives of individual views, we get the following joint

maximization problem for two graphs:

$$\max_{\substack{\mathbf{U}^{(v)}\in\mathbb{R}^{n\times k}\\ \mathbf{U}^{(w)}\in\mathbb{R}^{n\times k}}} tr\left(\mathbf{U}^{(v)^T}\mathcal{L}^{(v)}\mathbf{U}^{(v)}\right) + tr\left(\mathbf{U}^{(w)^T}\mathcal{L}^{(w)}\mathbf{U}^{(w)}\right) + \lambda\, tr\left(\mathbf{U}^{(v)}\mathbf{U}^{(v)^T}\mathbf{U}^{(w)}\mathbf{U}^{(w)^T}\right) \quad \text{s.t.} \quad \mathbf{U}^{(v)^T}\mathbf{U}^{(v)} = I,\ \mathbf{U}^{(w)^T}\mathbf{U}^{(w)} = I \tag{4.3}$$

The hyperparameter λ trades-off the spectral clustering objectives and the spectral embedding (dis)agreement term. The joint optimization problem given by Equation 4.3 can be solved using alternating maximization w.r.t. $\mathbf{U}^{(v)}$ and $\mathbf{U}^{(w)}$. For a given $\mathbf{U}^{(w)}$, we get the following optimization problem in $\mathbf{U}^{(v)}$:

$$\max_{\mathbf{U}^{(v)}\in\mathbb{R}^{n\times k}} tr\left\{\mathbf{U}^{(v)^T}\left(\mathcal{L}^{(v)} + \lambda\mathbf{U}^{(w)}\mathbf{U}^{(w)^T}\right)\mathbf{U}^{(v)}\right\}, \quad \text{s.t.} \quad \mathbf{U}^{(v)^T}\mathbf{U}^{(v)} = I. \tag{4.4}$$

This is a standard spectral clustering objective on view v with graph Laplacian $\mathcal{L}^{(v)} + \lambda\mathbf{U}^{(w)}\mathbf{U}^{(w)^T}$. This can be seen as a way of combining kernels or Laplacians. The difference from standard kernel combination (kernel addition, for example) is that the combination is adaptive since $\mathbf{U}^{(w)}$ keeps getting updated at each step, as guided by the clustering algorithm. The solution $\mathbf{U}^{(v)}$ is given by the top-k eigenvectors of this modified Laplacian. Since the alternating maximization can make the algorithm stuck in a local maximum (Niu et al., 2010), it is important to have a sensible initialization. If there is no prior information on which view is more *informative* about the clustering, we can start with any of the views. However, if we have some a priori knowledge on this, we can start with the graph Laplacian $\mathcal{L}^{(w)}$ of the more informative view and initialize $\mathbf{U}^{(w)}$. The alternating maximization is carried out after this until convergence. Note that one possibility could be to regularize directly on the eigenvectors $\mathbf{U}^{(v)}$'s and make them close to each other (e.g., in the sense of the Frobenius norm of the difference between $\mathbf{U}^{(v)}$ and $\mathbf{U}^{(w)}$). However, this type of regularization could be too restrictive and could end up shrinking the hypothesis space of feasible clusterings too much, thus ruling out many valid clusterings.

For fixed λ and n, the joint objective of Eq. 4.3 can be shown to be bounded from above by a constant. Since the objective is non-decreasing with the iterations, the algorithm is guaranteed to converge. In practice, the convergence is monitored by the difference in the value of the objective between consecutive iterations, and stop when the difference falls below a minimum threshold of $\epsilon = 10^{-4}$. In all the experiments, it converges within less than 10 iterations. Note that we can use either $\mathbf{U}^{(v)}$ or $\mathbf{U}^{(w)}$ in the final k-means step of the spectral clustering algorithm. In the experiments, a marginal difference in the clustering performance is noticed depending on which $\mathbf{U}^{(\cdot)}$ is used in the final step of k-means clustering.

4.1.2 Extension to Multiple Views

The co-regularized spectral clustering proposed in the previous section can be naturally extended for more than two views. This can be done by employing pair-wise co-regularizers in the objective function of Eq. 4.3. For m number of views, we have

$$\max_{\mathbf{U}^{(1)},\mathbf{U}^{(2)},\ldots,\mathbf{U}^{(m)}\in\mathbb{R}^{n\times k}} \sum_{v=1}^{m} tr\left(\mathbf{U}^{(v)^T}\mathcal{L}^{(v)}\mathbf{U}^{(v)}\right) + \lambda \sum_{\substack{1\le v,w\le m\\ v\neq w}} tr\left(\mathbf{U}^{(v)}\mathbf{U}^{(v)^T}\mathbf{U}^{(w)}\mathbf{U}^{(w)^T}\right), \quad \text{s.t.} \quad \mathbf{U}^{(v)^T}\mathbf{U}^{(v)} = I,\ \forall\ 1 \le v \le V \tag{4.5}$$

A common λ for all pair-wise co-regularizers is used for simplicity of exposition, however different λ's can be used for different pairs of views. Similar to the two-view case, it can be optimized by alternating maximization cycling over the views. With all but one $\mathbf{U}^{(v)}$ fixed, we have the following optimization problem:

$$\max_{\mathbf{U}^{(v)}} tr\left\{\mathbf{U}^{(v)^T}\left(\mathcal{L}^{(v)} + \lambda \sum_{\substack{1\leq w\leq m,\\ w\neq v}} \mathbf{U}^{(w)}\mathbf{U}^{(w)^T}\right)\mathbf{U}^{(v)}\right\}, \quad \text{s.t.} \quad \mathbf{U}^{(v)^T}\mathbf{U}^{(v)} = I \tag{4.6}$$

All $\mathbf{U}^{(v)}$, $2 \leq v \leq m$ are initialized by solving the spectral clustering problem for single views. The objective of Eq. 4.6 is solved for $\mathbf{U}^{(1)}$ given all other $\mathbf{U}^{(v)}$, $2 \leq v \leq m$. The optimization is then cycled over all views while keeping the previously obtained $\mathbf{U}^{(\cdot)}$'s fixed.

4.1.3 Centroid-Based Co-regularization

In this section, an alternative regularization scheme is presented that regularizes each view-specific set of eigenvectors $\mathbf{U}^{(v)}$ towards a common centroid $\mathbf{U}^*$ (akin to a *consensus* set of eigenvectors) . In contrast with the pairwise regularization approach which has $\binom{m}{2}$ pairwise regularization terms, where m is the number of views, the centroid based regularization scheme has m pairwise regularization terms. The objective function can be written as:

$$\begin{aligned}\max_{\mathbf{U}^{(1)},\mathbf{U}^{(2)},\ldots,\mathbf{U}^{(m)},\mathbf{U}^*\in\mathbb{R}^{n\times k}} &\sum_{v=1}^{m} tr\left(\mathbf{U}^{(v)^T}\mathcal{L}^{(v)}\mathbf{U}^{(v)}\right) + \sum_v \lambda_v tr\left(\mathbf{U}^{(v)}\mathbf{U}^{(v)^T}\mathbf{U}^*\mathbf{U}^{*^T}\right), \\ &\text{s.t.} \quad \mathbf{U}^{(v)^T}\mathbf{U}^{(v)} = I, \ \forall\ 1 \leq v \leq V, \quad \mathbf{U}^{*^T}\mathbf{U}^* = I\end{aligned} \tag{4.7}$$

This objective tries to balance a trade-off between the individual spectral clustering objectives and the agreement of each of the view-specific eigenvectors $\mathbf{U}^{(v)}$ with the consensus eigenvectors $\mathbf{U}^*$. Each regularization term is weighted by a parameter λ_v specific to that view, where λ_v can be set to reflect the importance of view v.

Just like for Equation 4.6, the objective in Equation 4.7 can be solved in an alternating fashion optimizing each of the $\mathbf{U}^{(v)}$'s one at a time, keeping all other variables fixed, followed by optimizing the consensus $\mathbf{U}^*$, keeping all the $\mathbf{U}^{(v)}$'s fixed.

It is easy to see that with all other view-specific eigenvectors and the consensus $\mathbf{U}^*$ fixed, optimizing $\mathbf{U}^{(v)}$ for view v amounts to solving the following:

$$\max_{\mathbf{U}^{(v)}\in\mathbb{R}^{n\times k}} tr\left(\mathbf{U}^{(v)^T}\mathcal{L}^{(v)}\mathbf{U}^{(v)}\right) + \lambda_v tr\left(\mathbf{U}^{(v)}\mathbf{U}^{(v)^T}\mathbf{U}^*\mathbf{U}^{*^T}\right), \quad \text{s.t.} \quad \mathbf{U}^{(v)^T}\mathbf{U}^{(v)} = I \tag{4.8}$$

which is nothing but equivalent to solving the standard spectral clustering objective for $\mathbf{U}^{(v)}$ with a modified Laplacian $\mathcal{L}^{(v)} + \lambda_v\mathbf{U}^*\mathbf{U}^{*^T}$. Solving for the consensus $\mathbf{U}^*$ requires solving the following objective:

$$\max_{\mathbf{U}^*\in\mathbb{R}^{n\times k}} \sum_v \lambda_v tr\left(\mathbf{U}^{(v)}\mathbf{U}^{(v)^T}\mathbf{U}^*\mathbf{U}^{*^T}\right), \quad \text{s.t.} \quad \mathbf{U}^{*^T}\mathbf{U}^* = I \tag{4.9}$$

Using the circular property of matrix trace, Equation 4.9 can be rewritten as:

$$\max_{\mathbf{U}^*\in\mathbb{R}^{n\times k}} tr\left\{\mathbf{U}^{*^T}\left(\sum_v \lambda_v\left(\mathbf{U}^{(v)}\mathbf{U}^{(v)^T}\right)\right)\mathbf{U}^*\right\}, \quad \text{s.t.} \quad \mathbf{U}^{*^T}\mathbf{U}^* = I \tag{4.10}$$

which is equivalent to solving the standard spectral clustering objective for $\mathbf{U}^*$ with a modified Laplacian $\sum_v \lambda_v \left(\mathbf{U}^{(v)}\mathbf{U}^{(v)^T}\right)$. In contrast with the pairwise co-regularization approach of Section 4.1.1 which computes optimal view specific eigenvectors $\mathbf{U}^{(v)}$'s, which finally need to be combined (e.g., via column-wise concatenation) before running the k-means step, the centroid-based co-regularization approach directly finds an optimal $\mathbf{U}^*$ to be used in the k-means step. One possible downside of the centroid-based co-regularization approach is that noisy views could potentially affect the optimal $\mathbf{U}^*$ as it depends on all the views. To deal with this, careful selection of the weighing parameter λ_v is required. If it is *a priori* known that some views are noisy, then it is advisable to use a small value of λ_v for such views, so as to prevent them from adversely affecting $\mathbf{U}^*$.

4.2 When will co-regularization help in spectral clustering?

In this section, I will discuss some example scenarios to better understand when co-regularization can be expected to help in spectral clustering. I also discuss some negative examples for which co-regularization does not help. The examples discussed below are same as those constructed in Sec. 3.3.

4.2.1 Correlated Features

Let us consider the example and experiment constructed in Sec. 3.3.1. This example simulates the scenario when we have correlated features across views given the cluster label. The plot of normalized mutual information (NMI) versus the correlation parameter ρ is shown in Fig. 4.1. (Please refer to Sec. 3.3.1 for definition of ρ.) A higher value of NMI indicates better clustering accuracy. The red and blue plots show the NMI obtained with best single view and with co-regularized spectral clustering algorithm, respectively. **The improvement obtained with co-regularization decreases with the increase in the parameter ρ.** This is not unexpected: a negative correlation between features in the two views in this case ensures that if a sample is more confusable with the other cluster in one view, it will be less confusable with the other cluster in the second view. Thus each view provides complementary information that helps improve the clustering. Totally uncorrelated features ($\rho = 0$) also help co-regularization improve beyond the best single view. On the other hand, when ρ is high, both views provide similar information and we do not gain much by the use of co-regularization. These observations are same as those for co-trained spectral clustering.

4.2.2 Complementary corruption in the similarity graphs

Let us consider the example and experiment constructed in Sec. 3.3.2. We saw in the last chapter that co-training was able to leverage on the fact that the noise was complementary in the two similarity matrices and was able to improve performance significantly beyond the performance of best single view. On the other hand, the co-regularized spectral clustering is *not* able to improve the NMI beyond that of best single view. It can be attributed to the fact that the co-regularized algorithm alters the graph Laplacians in an additive fashion and hence has a similar behavior as "similarity addition" (adding the two similarity matrices and doing spectral clustering on the

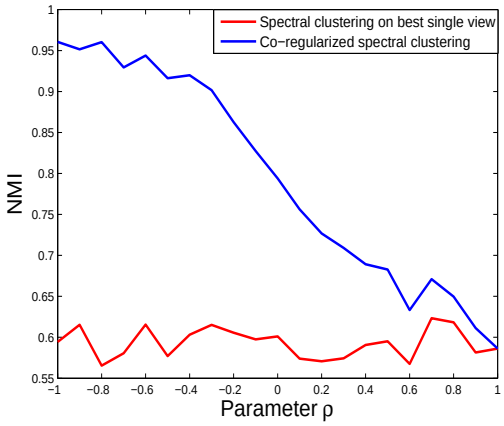

Figure 4.1: NMI scores for best single view and for multiview co-trained spectral clustering vs the feature correlation ρ

resultant matrix) in this case which is also not able to improve performance beyond that of best single view.

4.2.3 Low quality or contradictory individual views

Let us consider the examples and experiments constructed in Sec. 3.3.3. Here I test the assumptions of sufficiency (i.e., each view should be sufficient on its own to give a reasonably good clustering) and compatibility (the target clusterings in both views have same cluster assignments for co-occurring features with high probability) for co-regularized spectral clustering. Consider the first scenario where one of the similarity matrices does not have any useful information for clustering. The similarity matrices in the two views are given as follows.

$$K^{(1)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}, \qquad K^{(2)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}$$

I assume a ground truth clustering that first $\frac{n}{2}$ points belong to one cluster and the last $\frac{n}{2}$ points belong to the second cluster. The matrix $K^{(2)}$ has equal similarity for all pairs of points and does not have any useful information for clustering. For this case, the first view gives a perfect NMI of 1 and the second view gives extremely low NMI of the order of 10^{-4}. While the co-trained spectral clustering of the last chapter gave an NMI of 0.35 for this data, the co-regularized spectral clustering algorithm gives a perfect NMI of 1. This example illustrates that the uninformative second view does not affect co-regularized spectral clustering. Explicit tying of the eigenvector subspaces (UU^T) is able to overcome the effect of an uninformative Laplacian. Having uniformly distributed random numbers in $K^{(2)}$ instead of all 1's also gives the same result.

As a second example, consider the following similarity matrices for the two views that indicate contradictory clusterings.

$$K^{(1)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} \end{pmatrix}, \qquad K^{(2)} = \begin{pmatrix} \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} \\ \mathbf{0}_{\frac{n}{2}\times\frac{n}{4}} & \mathbf{1}_{\frac{n}{2}\times\frac{n}{2}} & \mathbf{0}_{\frac{n}{2}\times\frac{n}{4}} \\ \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}\times\frac{n}{2}} & \mathbf{1}_{\frac{n}{4}\times\frac{n}{4}} \end{pmatrix}$$

These similarity matrices indicate significantly different clustering assignments. In this case, the co-trained spectral clustering algorithm oscillates heavily across its iterations and does not help. The centroid version of co-regularized spectral clustering algorithm converges to an NMI value of 0.35 (assuming a ground truth clustering of first $\frac{n}{2}$ points belonging to one cluster and the last $\frac{n}{2}$ points belonging to the other cluster), which lies between the perfect NMI of 1 obtained with $K^{(1)}$ and NMI of 0 obtained with $K^{(2)}$. This is worse than the baseline of best single view which is also expected since one of the similarity matrices admits a significantly different clustering.

4.3 Experiments

Both of the co-regularization based multi-view spectral clustering approaches are compared with the same baselines as in previous chapter. To distinguish between the results of our two co-regularization based approaches, in the tables containing the results, I use symbol "P" to denote the *pairwise* co-regularization method and symbol "C" to denote the *centroid* based co-regularization method. For datasets with more than two views, the number of views are explicitly mentioned in parentheses.

Here, experimental results on two synthetic and three real-world datasets are reported. For more experimental results, the reader is referred to Kumar et al. (2011).

I give a brief description of the synthetic datasets here. The real datasets are same as used in the previous chapter.

- **Synthetic data 1:** The first synthetic dataset consists of two views and is generated in a manner akin to (Yi et al., 2005) which first chooses the cluster c_i each sample belongs to, and then generates each of the views $x_i^{(1)}$ and $x_i^{(2)}$ from a two-component Gaussian mixture model. These views are combined to form the sample $(x_i^{(1)}, x_i^{(2)}, c_i)$. We sample 1000 points from each view. The cluster means in view 1 are $\mu_1^{(1)} = (1\ \ 1)$, $\mu_2^{(1)} = (2\ \ 2)$, and in view 2 are $\mu_1^{(2)} = (2\ \ 2)$, $\mu_2^{(2)} = (1\ \ 1)$. The covariances for the two views are given below.

$$\Sigma_1^{(1)} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}, \Sigma_1^{(2)} = \begin{pmatrix} 0.3 & 0 \\ 0 & 0.6 \end{pmatrix}, \Sigma_2^{(1)} = \begin{pmatrix} 0.3 & 0 \\ 0 & 0.6 \end{pmatrix}, \Sigma_2^{(2)} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}$$

- **Synthetic data 2:** The second synthetic dataset consists of three views. Moreover, the features are correlated. Each view still has two clusters. Each view is generated by a two component Gaussian mixture model. The cluster means in view 1 are $\mu_1^{(1)} = (1\ \ 1)$, $\mu_2^{(1)} = (3\ \ 4)$; in view 2 are $\mu_1^{(2)} = (1\ \ 2)$, $\mu_2^{(2)} = (2\ \ 2)$; and in view 3 are $\mu_1^{(3)} = (1\ \ 1)$, $\mu_2^{(3)} = (3\ \ 3)$. The covariances for the three views are given below. The notation $\Sigma_c^{(v)}$ denotes the parameter for c'th cluster in v'th view.

$$\Sigma_1^{(1)} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}, \quad \Sigma_1^{(2)} = \begin{pmatrix} 1 & -0.2 \\ -0.2 & 1 \end{pmatrix}, \quad \Sigma_1^{(3)} = \begin{pmatrix} 1.2 & 0.2 \\ 0.2 & 1 \end{pmatrix}$$
$$\Sigma_2^{(1)} = \begin{pmatrix} 0.3 & 0.2 \\ 0.2 & 0.6 \end{pmatrix}, \quad \Sigma_2^{(2)} = \begin{pmatrix} 0.6 & 0.1 \\ 0.1 & 0.5 \end{pmatrix}, \quad \Sigma_2^{(3)} = \begin{pmatrix} 1 & 0.4 \\ 0.4 & 0.7 \end{pmatrix}$$

Here, I report only normalized mutual information (NMI) as the clustering quality evaluation measure, which gives the mutual information between obtained clustering and the true clustering

Method	Synth data 1	Synth data 2	Reuters	Handwritten	BBC
Best Single View	0.267 (0.0)	0.898 (0.0)	0.287 (0.019)	0.641 (0.008)	0.478 (0.001)
Feature Concat	0.294 (0.0)	0.923 (0.0)	0.298 (0.020)	0.619 (0.015)	0.512 (0.013)
Kernel Addition	0.339 (0.0)	0.973 (0.0)	0.323 (0.021)	0.744 (0.030)	0.506 (0.008)
Kernel Product	0.277 (0.0)	0.959 (0.0)	0.123 (0.010)	0.754 (0.026)	0.522 (0.025)
CCA	0.330 (0.0)	0.932 (0.0)	0.147 (0.003)	0.682 (0.019)	0.214 (0.001)
Min-Disagreement	0.313 (0.0)	0.936 (0.0)	0.342 (0.024)	0.745 (0.024)	0.794 (0.033)
Co-trained (2)	0.375 (0.0)	0.981 (0.0)	0.373 (0.012)	0.765 (0.031)	0.841 (0.000)
Co-trained (3)	–	0.989 (0.0)	0.388 (0.007)	–	–
Co-regularized (P) (2)	0.378 (0.0)	0.981 (0.0)	0.375 (0.02)	0.759 (0.031)	0.625 (0.006)
Co-regularized (P) (3)	–	0.989 (0.0)	0.376 (0.01)	–	–
Co-regularized (C) (2)	0.367 (0.0)	0.955 (0.0)	0.360 (0.025)	0.768 (0.025)	0.613(0.001)
Co-regularized (C) (3)	–	0.989 (0.0)	0.351 (0.021)	–	–

Table 4.1: NMI results on various datasets for different baselines and the proposed approaches. Numbers in parentheses are the std. deviations. The numbers (2), (3) and (4) indicate the number of views used in our co-regularized spectral clustering approach. Other multi-view baselines were run with maximum number of views available (or maximum number of views they can handle). Letters (P) and (C) indicate pairwise and centroid based regularizations respectively.

normalized by the cluster entropies. NMI ranges between 0 and 1 with higher value indicating closer match to the true clustering. Gaussian kernel is used for computing the graph similarities in all the experiments, unless mentioned otherwise. The standard deviation of the kernel is taken equal to the median of the pair-wise Euclidean distances between the data points. In the experiments, the co-regularization parameter λ is varied from 0.0005 to 0.002 and the best result is reported (λ is kept same for all views; one can however also choose different λ's based on the importance of individual views). The range of λ values is explored more exhaustively later in this Section where it is shown that the proposed approach outperforms other baselines for a wide range of λ. In the results table, the numbers in the parentheses are the standard deviations of the performance measures obtained with 20 different runs of k-means with random initializations.

4.3.1 Results

The results for all datasets are shown in Table 4.1. For two-view synthetic data (Synthetic Data 1), both the co-regularized spectral clustering approaches outperform all the baselines by a significant margin, with the pairwise approach doing marginally better than the centroid-based approach. The closest performing approaches are kernel addition and CCA. For synthetic data, order-2 polynomial kernel based kernel-CCA gives best performance among all CCA variants, while Gaussian kernel based kernel-CCA performs poorly. I do not report results for Gaussian kernel CCA here. All the multi-view baselines outperform the single view case for the synthetic data.

For three-view synthetic data (Synthetic Data 2), it can be seen that simple feature concatenation does not help much. In fact, it reduces the performance when the third view is added, so we report the performance with only two views for feature concatenation. Kernel addition with three views gives a good improvement over single view case. As compared to other baselines (with two views), both the co-regularized spectral clustering approaches with two views perform better. For

both approaches, addition of third view also results in improving the performance beyond the two view case.

For the document clustering results on Reuters multilingual data, English and French languages are used as the two views. On this dataset too, both the propsoed approaches outperform all the baselines by a significant margin. The next best performance is attained by minimum-disagreement spectral clustering (de Sa, 2005) approach. It should be noted that CCA and element-wise kernel product performances are worse than that of single view.

For UCI Handwritten digits dataset, quite a few approaches including kernel addition, element-wise kernel multiplication, and minimum-disagreement are close to both of our co-regularized spectral clustering approaches. It can be also be noted that feature concatenation actually performs worse than single view on this dataset. For BBC data, the proposed co-regularized approaches perform better than other baselines except de Sa (2005) and co-trained spectral clustering approach of previous chapter.

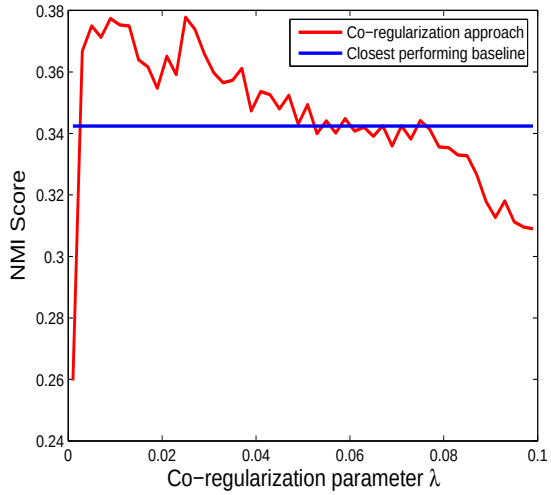

Figure 4.2: NMI scores of Co-regularized Spectral Clustering as a function of λ for **Reuters multilingual data**

Next, I show experiments with various values of co-regularization parameter λ observing its effect on the clustering performance. The reported results are for the pairwise co-regularization approach. Similar trends were observed for the centroid-based co-regularization approach. Fig. 4.3.1 shows the plot for Reuters multilingual data. The NMI score shoots up right after λ starts increasing from 0 and reaches a peak at $\lambda = 0.01$. After reaching a second peak at about 0.025, it starts decreasing and hovers around the second best baseline (Minimizing-disagreement in this case) for a while. The NMI becomes worse than the second best baseline after $\lambda = 0.075$. These results indicate that although the performance of our algorithms depends on the weighing parameter λ, it is reasonably stable across a wide range of λ. For more experiments on the robustness with respect λ, please see Kumar et al. (2011).

4.4 Discussion

This chapter presented a multi-view clustering approach in the framework of spectral clustering. The approach uses the philosophy of co-regularization to make the clusterings in different views agree with each other. Co-regularization idea has been used in the past for semi-supervised learning problems. To the best of my knowledge, this is the first work to apply the idea to the problem of unsupervised learning, in particular to spectral clustering. The co-regularized spectral clustering has a joint optimization function for spectral embeddings of all the views. An alternating maximization framework reduces the problem to the standard spectral clustering objective which is efficiently solvable using state-of-the-art eigensolvers. The approaches of this chapter can be extended to the missing data scenario. It can also be applied to dimensionality reduction similar to kernel PCA (Smola & Muller, 1999) and Laplacian eigenmaps (Belkin & Niyogi, 2003).

Part II

Supervised Learning with Multiple Kernels

Chapter 5

Multiple Kernel Learning via Binary Classification

In this chapter, I consider the problem of learning with multiple set of similarities in the context of classification. The similarities used in this chapter are positive (semi)definite kernel functions that are used in kernel methods (e.g., SVM, kernel ridge regression). The goal of this chapter is learn a "good" linear combination of the given multiple kernel functions so that the final kernel performs well on the SVM classification task.

Here, I introduce TS-MKL, a general approach to Two-Stage Multiple Kernel Learning that encompasses the previous work based on target alignment (Cortes et al., 2010b) as special cases. The kernel learning problem is formulated as a standard linear classification problem in a new instance space. In this space, any linear classifier with weights $\boldsymbol{\mu}$ directly corresponds to a linear combination of base kernels with weights $\boldsymbol{\mu}$. To avoid confusions, let us denote this new instance space as the *K-space*, and a classifier in the *K-space* as a *K-classifier*. Thus the problem of finding a "good" kernel combination reduces to finding a "good" linear classifier in the K-space, a very familiar problem. One big advantage of this approach is that one can easily adapt techniques from binary classification to solve the MKL problem. For instance, one can use familiar and well understood max-margin methods to obtain better performing MKL algorithms, or take advantage of the recent advances in large scale learning to scale up and/or parallelize the MKL implementations. For the results presented in this chapter, K-classifiers are learned (and hence kernels) by training L_2 regularized linear SVMs with positive weights using the stochastic projected sub-gradient descent method from Pegasos Shalev-Shwartz et al. (2007).

On the theoretical side, a finite sample generalization bound is shown for the original classification task in terms of the expected hinge loss and the margin of a K-classifier in the K-space. This justifies the proposed approach of training a K-classifier that has low hinge loss and high margin in the K-space in order to learn a good kernel for the original classification problem. To the best of my knowledge, this result represent the first finite sample bound for two-stage kernel learning, improving on previous bounds that were only asymptotic. A concentration bound for the expected hinge loss of a K-classifier is also given.

On the empirical side, a comprehensive evaluation on two object recognition datasets (Caltech 101 and 256), three bioinformatics datasets (Psort+, Psort-, Plant) and four UCI datasets is shown. On all these datasets the proposed method performs better than, or the same as target alignment, showing that choosing a better K-classifier is beneficial. The proposed method also fares well against one-stage multiple kernel learning approaches significantly outperforming them on Caltech-101 and Caltech-256 and being essentially tied on the others.

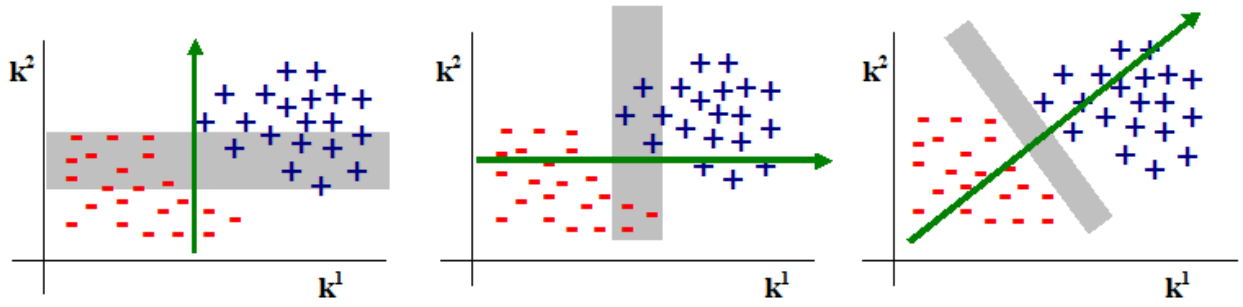

Figure 5.1: The K-space for two base kernels ($p = 2$). Points represent positive and negative K-examples $z_{xx'}$. The coordinates are the values of $K_1(x, x')$ and $K_2(x, x')$.

5.1 Method

Let us consider a classification problem where instances (x, y) are drawn from a distribution P over $\mathcal{X} \times \mathcal{Y}$, with $\mathcal{Y}$ a finite discrete set of labels. Let us assume that we have access to p positive semi-definite (PSD) base kernel functions $K_1, \cdots, K_p$ with $K_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The goal is to learn a combination of these kernel functions that is itself a positive semi-definite function and is a "good" kernel for the classification task at hand. To achieve this, let us define a new *binary* classification problem over a new instance space $\{(z_{xx'}, t_{yy'})|((x, y), (x', y')) \sim P \times P\} \subset \mathbb{R}^p \times \{\pm 1\}$ where

$$\begin{aligned} z_{xx'} &= \big(K_1(x, x'), \cdots, K_p(x, x')\big) \\ t_{yy'} &= 2 \cdot \mathbf{1}\{y = y'\} - 1 \end{aligned} \tag{5.1}$$

Let us call this space the *K-space*, and call $z_{xx'}$ a *K-example* or *K-instance* and $t_{yy'}$ a *K-label*. Any function $h : \mathbb{R}^p \to \mathbb{R}$ in this space induces a similarity function $\tilde{K}_h$ between instances in the original space:

$$\tilde{K}_h(x, x') = h(z_{xx'}) = h(K_1(x, x'), \cdots, K_p(x, x'))$$

If $\tilde{K}_h$ is also positive semi-definite, hence a valid kernel, we say that h is a *K-classifier*. For example, all linear functions with positive coefficients (i.e. $h_\mu(z_{xx'}) = \mu \cdot z_{xx'}$ with $\mu \geq 0$) are K-classifiers with the induced kernels $\tilde{K}_\mu$ being linear combinations of the p base kernels. Figure 5.1 shows a toy example for the case of two base kernels. Each point in the figure is a labeled K-example $(z_{xx'}, t_{yy'})$ corresponding to a pair $(x, y), (x', y')$ of original instances. Note that the figure is drawn in K-space, not in input space. For a linear K-classifier h_μ, the value of its induced kernel for a parir of original instances, $\tilde{K}_\mu(x, x')$, is the projection of the corresponding K-example $z_{xx'}$ on the vector μ (represented by the green line). The left and center sub-figures show the cases where μ is $(0, 1)$ and $(1, 0)$ respecively. In both cases the induced kernel combination is suboptimal. The linear combination in the right sub-figure corresponds to $\mu = (1, 1)$ and is a good combination because the kernel values of pairs of instances in the same class are separated from the kernel values of pairs of instances in different classes.

The key insight behind the proposed method is that, if a K-classifier h is a good classifier in the K-space, then the induced kernel $\tilde{K}_h(x, x') = h(z_{xx'})$ will likely be positive when x and x' belong to the same class and negative otherwise. This makes $\tilde{K}_h$ a good kernel for the original classification task. This intuition is made more precise in Section 5.2 where a generalization bound is given that shows that a K-classifier that separates the positive and negative K-examples with high margin will indeed induce a kernel that allows learning a good classifier for the original task. Note that having a good K-classifier is a sufficient condition, not a necessary one. There can very well exist combinations of base kernels that do not correspond to a good K-classifier, but are good kernels

nevertheless. Unlike one-stage kernel learning approaches, the proposed method will not be able to find such combinations and it might miss on some good kernels. The results in Section 5.3, however, show that this does not seem to be the case in practice, as it consistently matched or exceeded the performance of one-stage MKL.

Thus the problem of learning a good kernel can be reduced to the problem of learning a good K-classifier in the newly defined K-space: given a training sample $(x_i, y_i)_{i=1}^n$ for the original classification task, construct a K-training set $(z_{ij}, t_{ij})_{1 \leq i \leq j \leq n}$ and learn a K-classifier h from this sample. Any learning algorithm can be used for learning h provided that the induced kernel can be guaranteed to be a valid PSD kernel[1].

In line with the majority of the MKL work, here we focus on learning linear K-classifiers, and hence linear combinations of base kernels. The results in Section 5.2 suggest that it is desirable to have a maximum margin K-classifier, so we will use L_2 regularized linear SVM to learn the K-classifier, and ensure that the induced kernel is PSD by constraining the weights to be positive. One could, however, use a sparsity promoting regularizer (e.g., L_1 penalty) if a sparse combination of kernels is desired.

The optimization problem for learning the kernel weights $\boldsymbol{\mu}$ is thus given by

$$\min_{\boldsymbol{\mu} \geq 0} \frac{\lambda}{2} ||\boldsymbol{\mu}||^2 + \frac{1}{\binom{n}{2} + n} \sum_{1 \leq i \leq j \leq n} [1 - t_{ij} \boldsymbol{\mu} \cdot \mathbf{z}_{ij}]_+ \tag{5.2}$$

where $[1 - s]_+ = \max\{0, 1 - s\}$ is the hinge loss.

The stochastic projected sub-gradient descent implemented in Pegasos Shalev-Shwartz et al. (2007) is used to optimize this objective, with an additional projection to the non-negative constraint set after every gradient step. Using a stochastic optimization method allows it to scale very well despite the quadratic number of K-examples: computation time is not directly dependent on the number of instances, linear in the number of base kernels, and independent of the number of classes. If needed, memory usage can be reduced through streaming tecshniques or on the fly construction of the K-examples.

5.1.1 Connection to Target Alignment

Previous two-stage kernel learning approaches Cristianini et al. (2001); Cortes et al. (2010b) learn a non-negative linear combination of base kernels that maximizes the *alignment* with the target kernel $K^{(t)}(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j$ on the training set. This is achieved by solving the optimization problem

$$\max_{\boldsymbol{\mu} \geq 0} \frac{\langle \sum_{l=1}^p \mu_l \mathbf{K}_l, \mathbf{K}^{(t)} \rangle}{|| \sum_{l=1}^p \mu_l \mathbf{K}_l ||_F}, \quad \text{s.t. } ||\boldsymbol{\mu}||_2 = 1, \tag{5.3}$$

where $\mathbf{A}$ is the Gram matrix of kernel A on the training set, $\langle \mathbf{A}, \mathbf{B} \rangle = tr(\mathbf{A}\mathbf{B}^T)$ and $||\mathbf{A}||_F^2 = tr(\mathbf{A}\mathbf{A}^T)$.

[1]One could drop the PSD requirement and use any classifier, even a non-linear one, to obtain a *similarity function* rather than a proper kernel. The theory of learning with similarity functions Balcan & Blum (2006) can be then applied to learn a classifier for the original task. Generalization bounds similar to the ones in Section 5.2 would also hold for this case.

The above optimization problem can be re-written in the terminology of K-examples as follows:

$$\max_{\boldsymbol{\mu}\geq 0} \frac{\boldsymbol{\mu}^T \left(\sum_{t_{ij}=1} \mathbf{z}_{ij} - \sum_{t_{ij}=-1} \mathbf{z}_{ij}\right)}{\sqrt{\boldsymbol{\mu}^T \left(\sum_{\forall i,j} \mathbf{z}_{ij}\mathbf{z}_{ij}^T\right) \boldsymbol{\mu}}}, \text{s.t. } ||\boldsymbol{\mu}||_2 = 1$$

When the base kernels are centered, as proposed in Cortes et al. (2010b), the denominator represents the overall standard deviation of the projections of the K-examples on the vector $\boldsymbol{\mu}$. Hence target alignment attempts to find a projection direction $\boldsymbol{\mu}$ that maximize the difference between the sums of the projections of the positive and negative K-examples, while minimizing the overall variance of the projected K-examples. This is very similar to using Fisher-LDA in the K-space, with non-negativity constraints on $\boldsymbol{\mu}$. In fact, viewing target alignment from this perspective, makes it clear that it implicitly makes the assumption that the data is homoscedastic (the positive and negative K-examples have the same covariance), which might not be appropriate in real applications.

5.1.2 Connection to Learning with Hyperkernels

The approach proposed here can also be cast in the framework of learning with hyperkernels Ong et al. (2005) which provides a general recipe for kernel learning and includes Multiple Kernel Learning as a special case. It introduces the notions of kernel *quality functional*, a measure of "goodness" of a kernel that depends on the training data, and *Hyper Reproducing Kernel Hilbert Space*, an RKHS over kernel functions that defines the class of kernels that can be learned. Once the desired Hyper-RKHS and quality functional are specified, one has to solve a semi-definite program (SDP) to optimize the quality functional regularized by the norm induced by the Hyper-RKHS.

When using an SVM as the K-classifier, TS-MKL can be put in the learning with hyperkernels framework by defining the Hyper-RKHS to be the set of non-negative linear combinations of base kernels, and the quality functional to be the hinge loss in K-space. Considering this specific setting has significant advantages: it enables the use of simple and well understood binary classification techniques to learn the kernel, it enables a theoretical analysis, and it allows a significantly more scalable implementation. Equally important, all these advantages do not seem to come at the cost of reduced performance, as it is still performing on par with or better than competing MKL techniques.

5.2 Theoretical Results

In this section we make the connection between the performance a K-classifier in the K-space and the performance on the original problem precise. Thus justifying the approach taken in this paper not only intuitively, but also from a theoretical standpoint. Specifically, we bound the generalization error of an SVM that uses the kernel induced by a K-classifier in terms of the expected hinge loss and the margin of the K-classifier in the K-space:

Theorem 5.2.1. *Let P be a distribution on $\mathcal{X}\times\{\pm 1\}$, $z_{xx'}$ and $t_{yy'}$ be as in Equation 5.1, h be a K-classifier, and R be a constant s.t. $h(z_{xx}) \leq R^2\ \forall x \in \mathcal{X}$. Let*

$$HL_{h,\gamma} = E_{((x,y),(x',y'))\in P\times P}\left[\left[1 - \frac{t_{yy'}h(z_{xx'})}{\gamma}\right]_+\right]$$

be the expected K-space hinge loss relative to margin γ of the K-classifier h. Then, with probability $1-\delta$, a classifier $\hat{f}$ with generalization error

$$P_{(x,y)}\left[\!\left[y\hat{f}(x) \leq 0\right]\!\right] \leq HL_{h,\gamma} + \mathcal{O}\left(\sqrt{\frac{R^4 \ln(1/\delta)}{\gamma^2 n}}\right)$$

can be learned efficiently from a training sample of n instances drawn IID from P.

The theorem follows from the two lemmas stated below. The first lemma shows that a K-classifier that has low expected hinge loss in the K-space will induce a "good" kernel. The second lemma shows that a good kernel allows for a classifier with low generalization error to be efficiently learned from a finite training sample. The following definition states formally what we mean by a good kernel Srebro (2007).[2]

Definition 5.2.1. *A kernel K is an (ϵ, γ) good kernel in hinge loss with respect to a distribution P on $\mathcal{X} \times \{\pm 1\}$ if there exist a classifier $w \in \mathcal{H}_K$ with $\|w\|_{\mathcal{H}_K} = 1$ s.t.*

$$E_{(x,y)}\left[\!\left[\left[1 - \frac{y\langle w, \phi(x)\rangle}{\gamma}\right]_+\right]\!\right] \leq \epsilon$$

where $\mathcal{H}_K$ is the Hilbert space and $\phi(\cdot)$ is the feature mapping corresponding to K.

Lemma 5.2.1. *Let P, h, $HL_{h,\gamma}$, R be as in Theorem 5.2.1. Then the $\tilde{K}_h$ is a $(HL_{h,\gamma}, \gamma/R)$ good kernel in hinge loss with respect to P.*

Proof. Let $w = E_{(x',y')}(y'\tilde{\phi}(x')) \in \mathcal{H}_{\tilde{K}_h}$. We have:

$$\begin{aligned}
\epsilon &= E_{(x,y),(x',y')}\left[\!\left[\left[1 - \frac{t_{yy'}h_{xx'}}{\gamma}\right]_+\right]\!\right] \\
&= E_{(x,y),(x',y')}\left[\!\left[\left[1 - \frac{yy'\tilde{K}(x,x')}{\gamma}\right]_+\right]\!\right] \\
&= E_{(x,y)}\left[\!\left[E_{(x',y')}\left[\!\left[\left[1 - \frac{yy'\tilde{K}(x,x')}{\gamma}\right]_+ \middle| (x,y)\right]\!\right]\right]\!\right] \\
&\qquad \text{(Jensen's inequality)} \\
&\geq E_{(x,y)}\left[\!\left[\left[1 - \frac{E_{(x',y')}\left[\!\left[yy'\langle\tilde{\phi}(x'), \tilde{\phi}(x)\rangle | (x,y)\right]\!\right]}{\gamma}\right]_+\right]\!\right] \\
&= E_{(x,y)}\left[\!\left[\left[1 - \frac{y\langle w/||w||_{\mathcal{H}}, \tilde{\phi}(x)\rangle}{\gamma/||w||_{\mathcal{H}}}\right]_+\right]\!\right]
\end{aligned}$$

To conclude the proof, we bound $\|w\|_{\mathcal{H}}$ by R:

$$\|w\|^2_{\mathcal{H}} = E_{(x,y)}\left[\!\left[y\tilde{\phi}(x)\right]\!\right] \cdot E_{(x',y')}\left[\!\left[y'\tilde{\phi}(x')\right]\!\right]$$

[2] A kernel that does not satisfy this definition is not necessarily a "bad" kernel. We just can not make any formal statements with respect to its performance.

$$
\begin{aligned}
&= E_{(x,y),(x',y')}\left[\!\left[yy'\tilde{K}(x,x')\right]\!\right] \\
&\leq \sqrt{E_{(x,y),(x',y')}\left[\!\left[y^2y'^2\right]\!\right] \cdot E_{(x,y),(x',y')}\left[\!\left[\tilde{K}^2(x,x')\right]\!\right]} \\
&= \sqrt{E_{(x,y),(x',y')}\left[\!\left[\tilde{K}^2(x,x')\right]\!\right]} \leq R^2
\end{aligned}
$$

□

Lemma 5.2.2. *Let K be an (ϵ, γ) good kernel in hinge loss, with $K(x,x) \leq R^2 \; \forall x \in \mathcal{X}$. Let $(x_i, y_i)_{i=1}^n$ be an IID training sample, and $\hat{f}(x) = \hat{w} \cdot \phi(x)$ with*

$$
\hat{w} = \underset{||w||_{\mathcal{H}_K} \leq 1}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left[1 - \frac{y_i w \cdot \phi(x_i)}{\gamma}\right]_+
$$

be a kernel classifier that minimizes the average hinge loss relative to γ on the training sample. Then, with probability at least $1-\delta$, we have:

$$
P_{(x,y)}\left[\!\left[y\hat{f}(x) \leq 0\right]\!\right] \leq \epsilon + \mathcal{O}\left(\sqrt{\frac{R^2 \ln(1/\delta)}{\gamma^2 n}}\right)
$$

Lemma 5.2.2 follows directly from Theorem 21 in Bartlett & Mendelson (2002).

Thus, in the case of learning a linear combination of kernels, the following generalization bound applies:

Corollary 5.2.1. *Let $h_\mu(z_{xx'}) = \mu \cdot z_{xx'}$ be a K-classifier with $\|\mu\|_2 = 1$. Then, with probability at least $1-\delta$, a classifier $\hat{f}$ with generalization error*

$$
P_{(x,y)}\left[\!\left[y\hat{f}(x) \leq 0\right]\!\right] \leq HL_{h_\mu,\gamma} + \mathcal{O}\left(\sqrt{\frac{p \ln(1/\delta)}{\gamma^2 n}}\right)
$$

can be learned efficiently from a training sample of n instances drawn IID from P.

Corollary 5.2.2. *Let $h_\mu(z_{xx'}) = \mu \cdot z_{xx'}$ be a K-classifier with $\|\mu\|_1 = 1$. Then, with probability at least $1-\delta$, a classifier $\hat{f}$ with generalization error*

$$
P_{(x,y)}\left[\!\left[y\hat{f}(x) \leq 0\right]\!\right] \leq HL_{h_\mu,\gamma} + \mathcal{O}\left(\sqrt{\frac{\ln(1/\delta)}{\gamma^2 n}}\right)
$$

can be learned efficiently from a training sample of n instances drawn IID from P.

Note that, unlike in the one-stage kernel learning case, the generalization bound in Theorem 5.2.1 is in terms of the expected hinge loss of the K-classifier not the training hinge loss. Recently, a generalization bound for the classification problem in the K-space was given by Kar (2013).

We can, however, prove a concentration bound for the expected hinge loss of a K-classifier. This is the analog of the concentration bounds for target alignment in Cortes et al. (2010b); Cristianini et al. (2001).[3]

[3]This is not a regular generalization bound as the K-classifier is not allowed to depend on the training sample.

Theorem 5.2.2. *Let P, h, $HL_{h,\gamma}$, R be as in Theorem 5.2.1. Let $(x_i, y_i)_{i=1}^n$ be an IID sample distributed according to P. Then the following inequality holds with probability at least $1-\delta$*

$$HL_{h,\gamma} \leq \frac{2}{n^2-n}\sum_{i<j}\left[1-\frac{t_{ij}h(z_{ij})}{\gamma}\right]_+ + \sqrt{\frac{2\left(1+\frac{R^2}{\gamma}\right)^2 \ln\frac{1}{\delta}}{n}}$$

Proof. We will prove the concentration bound using McDiarmid's inequality. Let

$$\begin{aligned} f((x_1,y_1),\cdots,(x_n,y_n)) = \\ = \frac{2}{n(n-1)}\sum_{1\leq i<j\leq n}\left[1-\frac{y_i y_j \tilde{K}(x_i,x_j)}{\gamma}\right]_+ \end{aligned}$$

Let (x_l', y_l') be a new sample drawn at random from P. We have

$$\begin{aligned} |f((x_1,y_1),...,(x_l,y_l),...,(x_n,y_n)) - \\ - f((x_1,y_1),...,(x_l',y_l'),...,(x_n,y_n))| \leq \\ \leq \frac{2}{n(n-1)}\left(\sum_{i=1}^{l-1}\left|\left[1-\frac{y_i y_l \tilde{K}(x_i,x_l)}{\gamma}\right]_+ - \right.\right. \\ \left.\left. - \left[1-\frac{y_i y' l \tilde{K}(x_i,x_l')}{\gamma}\right]_+\right|\right) + \\ + \frac{2}{n(n-1)}\left(\sum_{i=l+1}^{n}\left|\left[1-\frac{y_l y_i \tilde{K}(x_l,x_i)}{\gamma}\right]_+ \right.\right. \\ \left.\left. - \left[1-\frac{y_l' y_i \tilde{K}(x_l',x_i)}{\gamma}\right]_+\right|\right) \\ \leq \frac{2}{n}\left(1+\frac{R^2}{\gamma}\right) \end{aligned}$$

Where the last inequality comes from the fact that for any (x,y) and (x',y')

$$0 \leq \left[1-\frac{yy'\tilde{K}(x,x')}{\gamma}\right]_+ \leq 1+\frac{R^2}{\gamma}$$

Applying McDiarmid's inequality gives

$$\begin{aligned} P\left[\left[E\left[\left[f((x_1,y_1),\cdots,(x_n,y_n))\right]\right] - \right.\right. \\ \left.\left. - f((x_1,y_1),\cdots,(x_n,y_n)) \geq \epsilon_1\right]\right] \leq \\ \leq \exp\left(\frac{-n\epsilon_1^2}{2\left(1+\frac{R^2}{\gamma}\right)^2}\right) \end{aligned}$$

The statement of the theorem is obtained by equating the right side with δ, and observing that for any $i \neq j$

$$E_{(x_i,y_i),(x_j,y_j)}\left[\left[\left[1-\frac{t_{ij}h(z_{ij})}{\gamma}\right]_+\right]\right] =$$

$$= E_{(x,y),(x',y')} \left[\!\!\left[\left[1 - \frac{t_{yy'} h(z_{x,x'})}{\gamma} \right]_+ \right]\!\!\right]$$

which implies

$$E\left[\!\left[f((x_1, y_1), \cdots, (x_n, y_n)) \right]\!\right] =$$
$$= E_{(x,y),(x',y')} \left[\!\!\left[\left[1 - \frac{t_{yy'} h(z_{x,x'})}{\gamma} \right]_+ \right]\!\!\right]$$

□

5.3 Empirical Evaluation

The proposed method is evaluated on two object recognition datasets (Caltech-101 and Caltech-256), three bioinformatics datasets (Psort+, Psort- and Plant), and four UCI datasets (Sonar, Pima, Vertebral and Ionosphere). The method is compared with several baselines: best kernel, uniform combination of base kernels (Average), target alignment, and the one-stage MKL algorithms SILP Sonnenburg et al. (2006), SimpleMKL Rakotomamonjy et al. (2007), L_2-Norm MLK Kloft et al. (2011), and UFO-MKL Orabona & Jie (2011). For two-stage methods, LIBSVM Chang & Lin (2011) is used to train the data classifier and select the regularization parameter C via 4-fold cross-validation for all datasets except Caltech where it is fixed at 1000. On multi-class problems, a one-vs-rest SVM is used. For one-stage approaches other than UFO-MKL, C is selected as above and a one-vs-rest scheme is used for multi-class problems. For UFO-MKL, the joint multi-class formulation is used and α and C are searched over a bi-dimensional grid. Following Orabona & Jie (2011), the optimization is run for 20 epochs on UCI datasets, 30 epochs on Caltech-101 and 100 epochs on Caltech-256. All kernels used in the experiments are centered and standardized to have zero mean and unit variance in feature space.

5.3.1 Methodology for TS-MKL

To learn kernel combination weights $\boldsymbol{\mu}$ with TS-MKL, the objective in Eq. 5.2 is optimized using Pegasos Shalev-Shwartz et al. (2007) with an additional projection to the non-negative constraint set after each sub-gradient step. A batch size of 100 is used for each sub-gradient computation and 10^3 sub-gradient steps are run for UCI datasets and 10^5 for all others. Figure 5.2, plots the test data accuracy versus the number of gradient iterations on Caltech-101, showing that after 10^5 iterations the change in accuracy is minimal. For the bigger Caltech-256 there is also essentially no change after 10^5 iterations. Subsampling is used to balance the positive and negative K-examples.

To select the parameter λ, a single 80%-20% random split of the Pegasos training set is used and the λ with the lowest validation hinge loss[4]s selected[5]. The search grid for λ is taken to be in the range of 100 to 10^{-8} dividing in each step by 4. A big advantage of this selection scheme for λ is that it is completely independent from the data classifier that will ultimately use the learned kernel. This keeps the setup simple and avoids intricate multi-level multi-dimensional validation schemes across the parameters of the data classifier and the K-classifier. Fig. 5.2, shows the hinge-loss in K-space, the accuracy of the K-classifier, and the accuracy of the data classifier that uses

[4] i

[5] Since the K-examples are dependent, the training and validation set will not be fully independent. Nevertheless, this does not seem to negatively affect the performance.

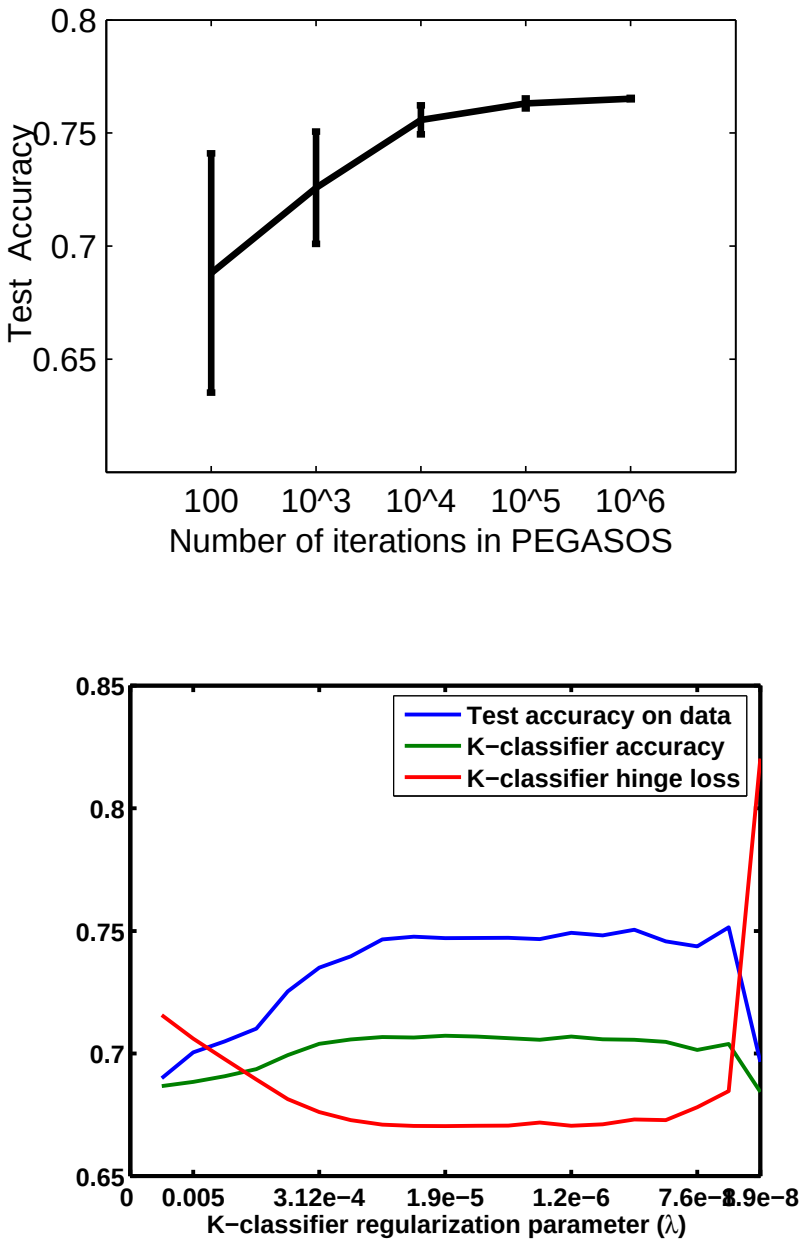

Figure 5.2: **Top:** Test data accuracy as a function of number of sub-gradient iterations in Pegasos. **Bottom:** Correlation between hinge loss (and accuracy) on K-examples and test data accuracy on Caltech-101.

the learned kernel, as a function of λ. The plot shows a clear correlation between hinge loss in K-space and data accuracy with the learned kernel. The data accuracy increases when the hinge loss in K-space decreases and vice versa. This experiment provides further emprical evidence for our theoretical results that show that a good K-classifier (having low hinge loss in K-space) corresponds to a good learned kernel.

After λ is selected, Pegasos is retrained on the full training set of K-examples. The obtained weight vector $\boldsymbol{\mu}$ is then used to linearly combine the base kernels, and the SVM data classifier is trained using this learned kernel with C selected as described above.

5.3.2 Caltech-101 and Caltech-256

Both these datasets contain pictures of objects and the task is to recognize the object category. Caltech-101 has 102 classes and Caltech-256 has 256 classes. Caltech-101 is perceived as an easier dataset than Caltech-256 in which images are not left-right aligned and there are more categories. The experimental setup used here is based on Gehler & Nowozin (2009) and use the same 39 base kernels and train test splits.

Results are reported using all 102 classes for Caltech-101 averaged over five splits. For Caltech-256, the results are for 256 classes (excluding the clutter category), for a single split. The performance measure used is mean prediction rate per class. The number of training images per class is varied in the range $5, 10, 15, 20, 25, 30$ for Caltech-101, and in the range $5, 10, 15, 20, 25, 30, 40, 50$ for Caltech-256. The number of test images used is up to 50 images per class for Caltech-101 and 25 images per class for Caltech-256. The regularization parameter for the data SVM, C, is fixed to 1000 for all methods[6].

The results for Caltech-101 and Caltech-256 are shown in Fig. 5.3.[7] On Caltech-101, the proposed approach yields a mean accuracy of 0.512, 0.630, 0.691, 0.725, 0.752, 0.772 for 5, 10, 15, 20, 25, 30 samples per class respectively. Comparing to UFO-MKL, the performance of TS-MKL is higher for 5 samples per class, and very similar for all other sample sizes. One-stage MKL methods using the one-vs-all multi-class scheme perform significantly worse and do not even outperform the average kernel until the training set has 25 samples per class. This is probably because data is too scarce to allow learning a separate kernel for each class. Target alignment performs a little better than the average kernel, but is still significantly worse than TS-MKL. Fig. 5.3 also shows the performance of LP-β (Gehler & Nowozin, 2009). The performance of TS-MKL and UFO-MKL is almost on par with LP-β, especially for larger sample sizes. While LP-β is similar in spirit to multiple-kernel learning, it is not a true kernel learning algorithm as it does not produce a kernel, but rather learns an ensemble of SVM classifiers, each of which is trained on an individual kernel. The best reported accuracy on Caltech-101 with 30 training samples per class is 82.5% (Bo et al., 2012).

On Caltech-256 dataset, TS-MKL performs better than all competing kernel learning baselines. It achieves 0.245, 0.320, 0.370, 0.426, 0.448, 0.475, 0.494 mean accuracy for 5, 10, 15, 20, 25, 30, 40, 50 training samples per class. This performance is significantly higher than the best results reported in the literature for 5, 10, and 15 training samples, after which it again performs on par with LP-β. On this dataset, UFO-MKL performance[8] is similar to that of the average kernel, while the rest of the one-stage MKL techniques perform worse. Exact target alignment is worst among all other

[6] $C = 1000$ is the best setting for the one-stage MKL algorithms (Gehler & Nowozin, 2009)

[7] The results for LP-β and MKL are taken from Gehler & Nowozin (2009).

[8] The UFO-MKL performance at 40 and 50 samples is missing because the code we are using runs out of memory.

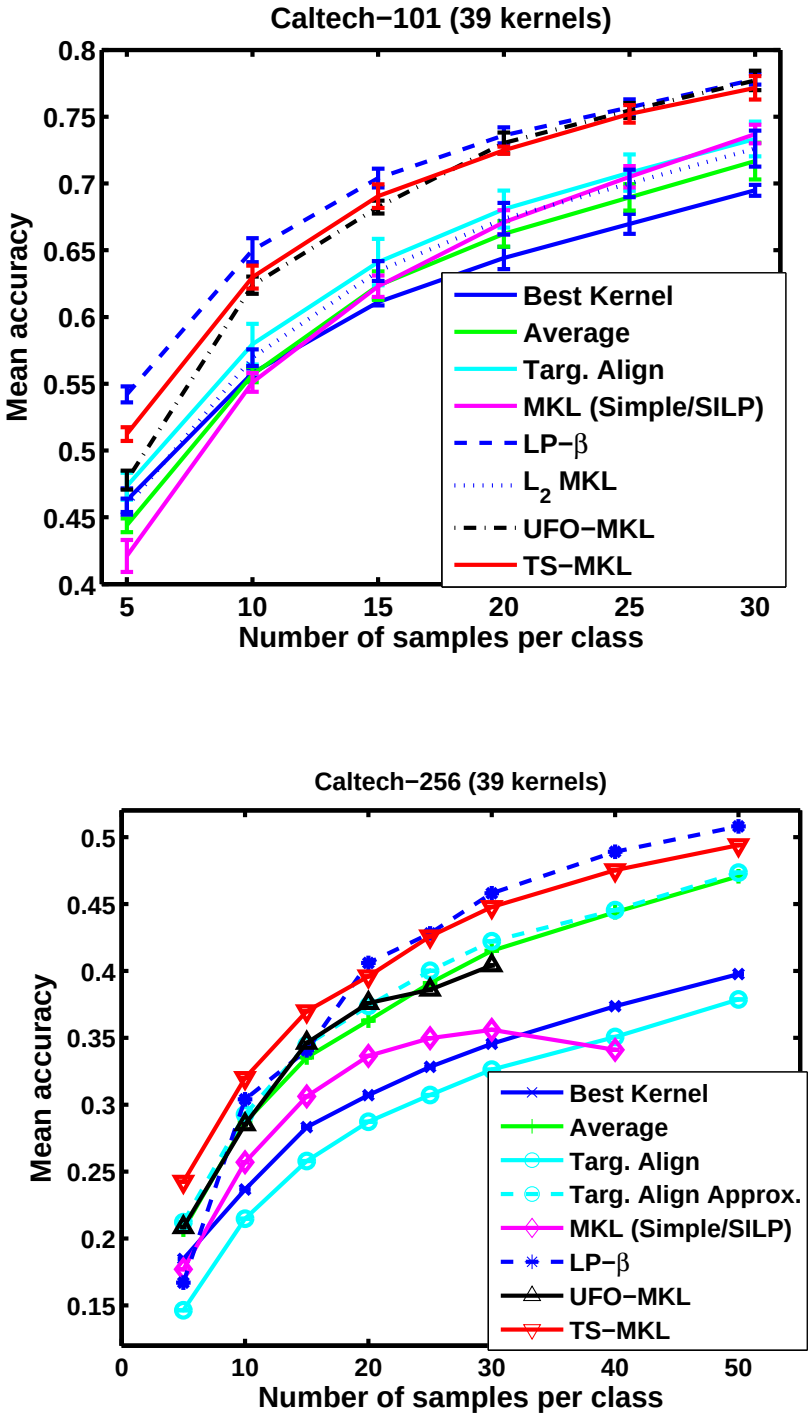

Figure 5.3: **Top:** Caltech-101 results: mean accuracy over all classes for different sample sizes, averaged over 5 splits. **Bottom:** Caltech-256 results: mean accuracy over all classes for different sample sizes

approaches, however approximate target alignment is able to at least match the performance of the average kernel. The best reported accuracy on Caltech-256 with 30 training samples per class is 50.7% (Bo et al., 2012).

	Psort+		Psort–		Plant
	Full test	Filtered	Full test	Filtered	Full test
Best Kernel	81.30(4.69)	86.26(4.96)	85.95(1.54)	91.53(1.04)	72.19(3.94)
Average	84.75(3.97)	89.48(4.97)	88.03(1.10)	93.95(1.14)	86.72(3.38)
Target Alignment	88.14(3.99)	92.82(3.99)	89.91(1.42)	95.22(1.33)	89.13(2.75)
MKL (SILP/Simple)	89.05(3.02)	93.89(3.37)	91.01(1.10)	96.01(1.51)	89.32(2.76)
MC-MKL	–	93.8	–	96.1	89.1
TS-MKL(Our Approach)	89.08(3.32)	93.50(2.74)	90.15(1.33)	95.63(1.31)	88.86(3.26)

Table 5.1: Average accuracy measures (%) over 10 splits for Psort+, Psort- and Plant datasets. Numbers in parentheses are the std. deviations. The accuracy measures for MC-MKL (Zien & Ong, 2007) are taken from their paper.

5.3.3 Bioinformatics datasets

The proposed method is evaluated on a problem relevant to cell-biology predicting: the subcellular localization of proteins, which is crucial in making inference about protein function and protein interactions. The experimental setup of Zien & Ong (2007) is followed with the same 69 kernels. The kernels used are: 2 kernels on phylogenetic trees, 3 kernels from BLAST E-values and 64 sequence motif kernels.

Experiments with three datasets are shown. The first two datasets are for the problem of bacterial protein locations Gardy et al. (2004). The Psort+ dataset has 541 data points with 4 classes and Psort– dataset has 1444 data points with 5 classes. Average F1 score over all classes over 10 random splits is reported for both these datasets as done in Zien & Ong (2007). The third dataset used is the original plant dataset of TargetP Emanuelsson et al. (2000), and has 940 examples with 4 classes. The performance measure of Matthew's Correlation Coefficient (MCC) is used following the evaluation in Zien & Ong (2007). Again, average MCC score over all 4 classes is reported.

The results are shown in Table 5.1. The papers that have used the Psort datasets in the past Gardy et al. (2004); Zien & Ong (2007), reported results after filtering out the most unsure predictions in the test set. For Psort+ and Psort-, about 15% and 13.3% of the test examples were filtered out respectively and the performance is reported only for the remaining predictions. The same procedure is followed here to be able to compare with these methods. Performance for full test set is also reported. On these datasets, all the kernel learning methods have similar performance, and are better than the best kernel and average kernel baselines. Multi-class multiple kernel learning (MC-MKL) of Zien & Ong (2007) is also close to TS-MKL and other baselines.

5.3.4 UCI datasets

Four UCI datasets are used: Sonar, Ionosphere, Pima and Vertebral (the three class version). For each of these datasets, two types of MKL experiments are performed. In first setting, we construct a total of 13 kernels on the full feature vectors: 9 Gaussian kernels ($e^{-\gamma||\mathbf{x}_i-\mathbf{x}_j||^2}$) with $\gamma = \{2^{-10}, 2^{-9}, \ldots, 2^{-2}\}$, 3 polynomial kernels of degree 2,3 and 4, and a linear kernel. In the

	Sonar		Ionosphere		Pima		Vertebral	
	p = 793	p = 13	p = 442	p = 13	p = 117	p = 13	p = 91	p = 13
Best Kernel	86.90(4.23)	86.90(4.24)	95.00(2.04)	95.00(2.04)	76.10(2.63)	76.10(2.63)	83.65(5.73)	83.67(5.73)
Average	85.00(4.5)	86.42(3.73)	92.00(2.87)	94.28(3.01)	76.82(2.76)	76.30(2.62)	81.58(5.92)	82.03(5.03)
Targ. Align	80.24(4.2)	85.47(3.26)	91.57(2.28)	94.42(2.17)	75.97(3.03)	76.82(3.15)	82.88(6.18)	80.90(4.18)
MKL(SILP/Simple)	85.23(5.11)	84.76(2.55)	92.54(1.56)	95.42(2.50)	75.71(3.28)	75.97(3.16)	82.72(4.16)	78.42(3.55)
L_2-MKL	86.42(4.05)	85.71(4.04)	91.85(1.51)	95.14(2.04)	75.45(2.31)	76.55(2.23)	79.68(4.84)	80.87(5.1)
UFO-MKL	82.85(6.7)	86.19(4.3)	91.85(2.86)	96.14(1.9)	74.28(2.47)	74.02(3.46)	79.16(6.57)	79.09(6.03)
TS-MKL(Our Approach)	86.43(3.9)	86.19(3.38)	92.43(1.18)	94.29(2.12)	75.78(3.02)	76.42(2.87)	82.82(5.63)	81.10(4.42)

Table 5.2: Average accuracy (%) over 10 random splits on UCI datasets. p denotes the number of base kernels. Numbers in parentheses are the std. deviations.

second setting, we augment these 13 kernels with another set of Gaussian, polynomial and linear kernels constructed on individual features of the data. The range of parameter γ for Gaussian and degree parameter for polynomial kernel is kept same as before. If the data has d features, we have total $13d+13$ kernels in the second setting. Average accuracy over 10 random $80\% - 20\%$ train-test splits is reported.

The results are shown in Table 5.2. On all these datasets, no kernel learning approach seems to improve performance over the straightforward baselines of best kernel and average kernel. Although further study is needed to reach a definite conclusion, these results seem to indicate that blindly using a kitchen sink of standard kernels is not beneficial if the goal is to combine these kernels using an MKL approach. This highlights the importance of evaluating MKL techniques using datasets like Caltech and PSORT, where the kernels have been carefully designed using domain knowledge to capture different, potentially useful, notions of similarity in the data.

5.3.5 Computational Efficiency

Since the number of K-examples is quadratic in the number of training instances, one might worry about the scalability of the TS-MKL method. This section compares the running time of TS-MKL with Target Alignment, and UFO-MKL (Ultra-Fast Optimization MKL) which is arguably the fastest one-stage MKL technique to date.

Table 5.3 shows the running times for the Sonar, Pima and Caltech 101 datasets. The running time is for a single run using the best setting of parameters (i.e. it does not include the time for parameter selection). For TS-MKL and Target Alignment we also show in parenthesis the time taken by the kernel learning stage alone, without the final data SVM, on Caltech-101. For Sonar, which has only 166 training samples, the running time of UFO-MKL and TS-MKL is comparable. However, on Pima, which has 614 samples, and on Caltech, which has 3060 samples and 102 classes, TS-MKL is more than an order of magnitude faster than UFO-MKL. This shows that, by taking advantage of the advances in large scale stochastic optimization, TS-MKL is not only able to gracefully handle the quadratic increase in the number of K-examples, but it is actually the fastest MKL method to date.

5.4 Discussion

Framing kernel learning as a standard classification problem in a properly defined instance space allows to easily adapt well understood classification techniques to obtain a scalable and high

	Sonar p = 793	Pima p = 117	Caltech 101 Train 30
Targ. Align	133	93.71	607(579)
UFO-MKL	3.018	17.97	387
TS-MKL	1.09	1.3977	34(6)

Table 5.3: Running time in seconds. In parenthesis we show the time taken by the kernel learning stage alone.

performing two-stage multiple kernel learning algorithm. The proposed approach is backed up by formal theoretical guarantees, and by empirical evaluation that shows it always outperforms or is on par with leading one-stage and two-stage kernel learning methods. This is a remarkable feat for a method that is quite simple and intuitive.

This new perspective on multiple kernel learning opens the door to a number of interesting questions to be addressed in subsequent research. A few examples are:

- **Learning kernel in scarce data conditions:** In many applications, we are faced with scarcity of labeled data but may have access to plenty of unlabeled data, which can be used to suitably regularize the hypothesis space and improve the prediction performance. In several other cases, we have multiple "related" prediction problems or tasks with a only a few labeled samples per task (Caruana, 1997). In these situations, it is advantageous to simultaneously learn these multiple tasks and use the relatedness structure of tasks to suitably regularize the joint hypothesis space (Bakker & Heskes, 2003; Jacob et al., 2008; Xue et al., 2007). Automatically learning the kernel function in these scarce data conditions through semi-supervised and multi-task multiple kernel learning is another interesting direction that can be pursued by transforming the problem into semi-supervised and multi-task learning of K-classifier.

- **Learning kernel for semi-supervised clustering:** In semi-supervised clustering and semi-supervised dimensionality reduction (e.g., kernel PCA), the supervision signal is usually given in terms of pairwise must-link and cannot-link constraints rather than labels for individual examples. The approach presented in the previous Chapter can be naturally extended to this setting as we just need to know whether a given pair of examples is from the same class or not.

Chapter 6

Learning Nonlinear Combination of Kernels

In the previous Chapter, an approach to learn a "good" linear combination of base kernels was presented that works by transforming the problem of multiple kernel learning into a binary classification problem in another space $\mathcal{Z}$ (referred as K-space). For p number of base kernels, the examples in the K-space (referred as K-examples) are constructed as $\mathbf{z}_{ij} = (K_1(\mathbf{x}_i, \mathbf{x}_j), K_2(\mathbf{x}_i, \mathbf{x}_j), \ldots, K_p(\mathbf{x}_i, \mathbf{x}_j))$ with labels $t_{ij} = 1$ if training examples $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to same class and -1 otherwise. Finding a good linear classifier $\mu : \mathcal{Z} \mapsto \mathbb{R}$ in K-space (referred as K-classifier) that assigns high value to positive K-examples and low value to negative K-examples is equivalent to finding a good linear combination of base kernels ($\boldsymbol{\mu}'\mathbf{z}_{ij} = \sum_{i=1}^{p} \mu_i K_i(\mathbf{x}_i, \mathbf{x}_j)$). This approach always outperformed or was on par with leading one-stage and two-stage kernel learning methods.

There are several research directions that can be pursued motivated by these ideas, some of which were mentioned in Sec. 5.4. A particularly appealing direction is to extend the framework proposed in the previous Chapter to learn a nonlinear combination of kernels. Even though there has been a significant amount of work on learning linear combinations of base kernels (Rakotomamonjy et al., 2007; Sonnenburg et al., 2006; Cortes et al., 2010a; Kloft et al., 2011; Bach, 2008; Zien & Ong, 2007; Sindhwani & Lozano, 2011), in many cases it has not been found to improve the empirical performance considerably beyond the simple baselines of kernel averaging and single kernel with parameter tuned by cross-validation (Gehler & Nowozin, 2008). One possible reason for this could be that the space of linear combination of the given base kernels is not rich enough to approximate the optimal kernel well. Investigating nonlinear combination of base kernels is a natural next step that I would pursue in this Chapter.

6.1 Related work

There have been a few studies on learning nonlinear kernel combinations (Ong et al., 2005; Tsang & Kwok, 2006; Varma & Babu, 2009; Cortes et al., 2009a). Ong et al. (2005) provide a general recipe for kernel learning with Multiple Kernel Learning as a special case. It introduces the notions of kernel quality functional, a measure of "goodness" of a kernel that depends on the training data, and Hyper Reproducing Kernel Hilbert Space, an RKHS over kernel functions that defines the class of kernels that can be learned. Once the desired Hyper-RKHS and quality functional are specified, one has to solve a semi-definite program (SDP) to optimize the quality functional regularized by the norm induced by the Hyper-RKHS. They show some encouraging empirical results on UCI datasets (UCI) with two proposed hyperkernels. However, the method is computationally

very expensive and does not scale well. It was later shown to be equivalent to a second-order cone program (SOCP) that improved on computational time (Tsang & Kwok, 2006), however it was still not suitable for large scale problems.

Varma & Babu (2009) proposed a multiple kernel learning approach that worked with product of RBF base kernels instead of linear combinations. It was mainly shown to be effective in doing feature selection. However, when all features were used, it did not show any empirical performance gains over the simple approach of assigning uniform weights to features chosen through cross-validation. Cortes et al. (2009a) proposed a method to learn polynomial combinations of kernels in the context of ridge regression that jointly learns the predictor and the kernel combination weights. They also reported a few empirical results on UCI datasets showing minor improvements over linear kernel combination in regression problems. However, a convincing empirical study showing definite improvements over linear kernel combinations was lacking.

6.2 Binary classification approach to nonlinear MKL

Existing approaches for learning nonlinear kernel combinations are either computationally too expensive to work with standard MKL datasets used in the previous Chapter (Ong et al., 2005; Tsang & Kwok, 2006), or do not give desired performance gains over simple baseline methods (Varma & Babu, 2009). In some cases, only a specific class of the nonlinear combinations is considered, e.g., polynomial combinations in regression setting (Cortes et al., 2009a) and products of RBF kernels (Varma & Babu, 2009).

The proposed MKL method in the previous Chapter gave encouraging results with linear kernel combinations, either outperforming or matching the leading MKL methods in terms of empirical performance. It also outperformed the previous fastest MKL methods in terms of computational efficiency (Table 5.3). Inspired by this, I extend the framework of previous Chapter to learn nonlinear kernel combinations. As described in the previous Chapter, a good K-classifier induces a good similarity function. Further, if the resulting similarity function is positive semidefinite then it is also a kernel function that can be directly used in kernel methods like SVM. Previous Chapter focused on learning linear combination of kernels by learning a linear K-classifier. In this Chapter, I extend this work to learn a nonlinear combination of kernels by learning a nonlinear K-classifier and study the empirical performance gains in comparison to linear combination of kernels on standard datasets.

Let us denote the nonlinear K-classifier by $h : \mathcal{Z} \mapsto \mathbb{R}$ that separates the positive and negative K-examples and induces the similarity measure $\tilde{K}_h(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{z}_{ij})$ where $\mathbf{z}_{ij}$ are the K-examples for all $1 \leq i, j \leq n$ (Eq. 5.1). There can be multiple ways to learn a nonlinear K-classifier. I study two types of nonlinear K-classifiers, one of these resulting in positive semidefinite kernel and the other resulting in indefinite similarity function.

6.2.1 Positive semidefinite kernel using polynomial kernel combination

If the resulting similarity function is positive semidefinite, it can be treated as a kernel and directly used in the kernel based learning algorithms. However, making sure the positive semidefiniteness of the resulting similarity ($\tilde{K}_h(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{z}_{ij})$) is a difficult task and heavily depends on the type of nonlinear K-classifier $h(\cdot)$ used and the way the K-examples are generated. Consider, for example, a polynomial K-classifier of degree two. If we explicitly enumerate the polynomial

combinations to construct the K-examples as follows

$$\mathbf{z}_{ij} = (\underbrace{K_1(\mathbf{x}_i, \mathbf{x}_j), \ldots, K_p(\mathbf{x}_i, \mathbf{x}_j)}_{p \text{ first-order terms}}, \underbrace{K_1(\mathbf{x}_i, \mathbf{x}_j)K_1(\mathbf{x}_i, \mathbf{x}_j),\ K_1(\mathbf{x}_i, \mathbf{x}_j)K_2(\mathbf{x}_i, \mathbf{x}_j), \ldots, K_p(\mathbf{x}_i, \mathbf{x}_i)K_p(\mathbf{x}_i, \mathbf{x}_j)}_{p^2 \text{ second-order terms}}), \tag{6.1}$$

and learn a linear K-classifier for these K-examples restricting the weights to be nonnegative, the induced kernel $\tilde{K}_h(\mathbf{x}_i, \mathbf{x}_j) = h(\mathbf{z}_{ij})$ will be positive semidefinite (element-wise product of positive semidefinite matrices is positive semidefinite). Let us call this method as **Polynomial TS-MKL**. However, if we simply use an SVM with polynomial kernel as our nonlinear K-classifier $h(\cdot)$, it is not clear how to guarantee positive semidefiniteness of $\tilde{K}_h(\mathbf{u}, \mathbf{v})$ in an efficient manner.

6.2.1.1 Polynomial kernel combination: a toy experiment

This toy experiment considers an example scenario where polynomial kernel combination will perform better than a linear combination. This two-class data is generated on two concentric circles in two dimensions as $\{\mathbf{x}_i = r[\cos(\theta_i),\ \sin(\theta_i)]\}_{i=1}^n$ with $r = 1$ for positive class and $r = 2$ for negative class. The angles θ_i are generated from a uniform distribution between 0 and 2π. Let us take the two linear kernels on individual features as our base kernels, i.e., $K_1(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_{i_1}\mathbf{x}_{j_1}$ and $K_1(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_{i_2}\mathbf{x}_{j_2}$.

Fig. 6.1 shows the data points (left) and the K-examples (middle) for same class (K-label $= 1$) and different class (K-label $= -1$). It is clear that positive and negative K-examples are not separated from each other along any linear direction. The right figure shows two dimensions $[K_1^2(\mathbf{x}_i, \mathbf{x}_j),\ [K_2^2(\mathbf{x}_i, \mathbf{x}_j)]$ of polynomial K-example (Eq. 6.1). The K-examples are more separated in this space. For linear kernel combination, we construct the K-examples as $\mathbf{z}_{ij} = [K_1(\mathbf{x}_i, \mathbf{x}_j),\ K_2(\mathbf{x}_i, \mathbf{x}_j)]$ and solve the following optimization problem to learn the kernel weights μ.

$$\min_{\boldsymbol{\mu} \geq 0} \frac{\lambda}{2}||\boldsymbol{\mu}||^2 + \frac{1}{\binom{n}{2} + n} \sum_{1 \leq i \leq j \leq n} [1 - t_{ij}\boldsymbol{\mu} \cdot \mathbf{z}_{ij}]_+ \tag{6.2}$$

where $[1 - s]_+ = \max\{0, 1 - s\}$ is the hinge loss.

For polynomial kernel combination, we construct the K-examples as

$$\mathbf{z}_{ij} = [K_1(\mathbf{x}_i, \mathbf{x}_j),\ K_2(\mathbf{x}_i, \mathbf{x}_j),\ K_1^2(\mathbf{x}_i, \mathbf{x}_j),\ K_2^2(\mathbf{x}_i, \mathbf{x}_j),\ K_1(\mathbf{x}_i, \mathbf{x}_j)K_2(\mathbf{x}_i, \mathbf{x}_j)],$$

and solve the above optimization problem to learn the five dimensional kernel weight vector μ, which is used to get the final kernel. The linear kernel combination gives significantly low training accuracy of 54%, as expected. On the other hand, using polynomial kernel combination, we are able to get 100% accuracy on the training set.

6.2.1.2 Polynomial kernel combination: empirical findings

This section tests the performance of Polynomial TS-MKL on real datasets. For UCI datasets, I construct linear kernels on individual features and use these as base kernels for the TS-MKL method. Table 6.1 reports the results obtained using linear TS-MKL and Polynomial TS-MKL (degree-2 and degree-3 expansion of K-examples). Polynomial TS-MKL of degree-2 improves the accuracy over its linear counterpart for all the datasets. Polynomial TS-MKL of degree-3 performs

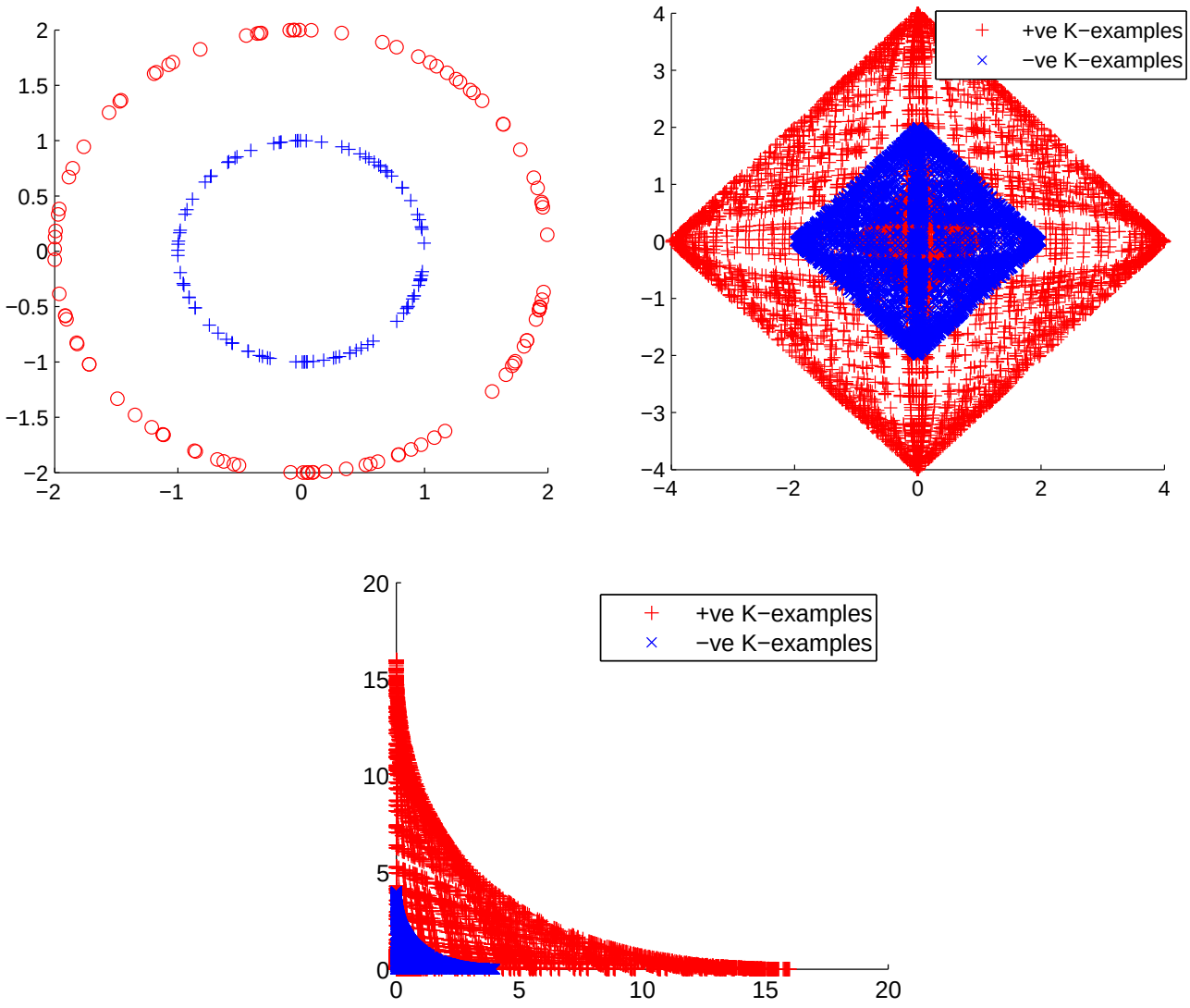

Figure 6.1: **Left:** Two circles data. **Middle:** K-examples $\mathbf{z}_{ij} = [K_1(\mathbf{x}_i, \mathbf{x}_j),\ K_2(\mathbf{x}_i, \mathbf{x}_j)]$. **Right:** Polynomial K-examples $\mathbf{z}_{ij} = [K_1^2(\mathbf{x}_i, \mathbf{x}_j),\ [K_2^2(\mathbf{x}_i, \mathbf{x}_j)]$.

	Sonar	Ionosphere	Pima	Vertebral
Average Kernel	74.52(5.9)	87.14(3.4)	75.84(2.5)	81.02(5.3)
TS-MKL Linear	73.10(3.4)	87.00(3.3)	76.43(2.1)	81.13(4.7)
TS-MKL Polynomial-2	83.57(2.9)	90.43(3.1)	76.56(2.8)	82.48(5.3)
TS-MKL Polynomial-3	82.10(5.9)	89.21(3.8)	74.83(2.6)	78.20(4.2)

Table 6.1: Average accuracy (%) over 10 random splits on UCI datasets. Numbers in parentheses are the std. deviations. TS-MKL Linear is the method proposed in Chapter 5. TS-MKL Polynomial-2 and Polynomial-3 learn linear K-classifiers on polynomial K-examples (Eq. 6.1) of degree 2 and degree 3 respectively and thus learn a polynomial kernel combination.

	Training samples per class					
	5	10	15	20	25	30
Average Kernel	44.40 (0.51)	55.69(0.57)	62.35(1.07)	66.22(0.95)	68.96(0.99)	71.68(1.38)
TS-MKL Linear	51.23 (0.51)	62.99(0.86)	69.05(0.88)	72.49(0.27)	75.21(0.66)	77.16(0.88)
TS-MKL Polynomial-2	50.75(0.41)	62.94(0.76)	67.71(0.88)	71.69(0.48)	74.87(0.81)	75.14(0.73)
TS-MKL Polynomial-3	49.36(0.63)	60.29(0.82)	67.23(0.75)	70.09(0.74)	73.61(0.93)	74.31(0.93)

Table 6.2: Average accuracy (%) over 5 random splits on Caltech-101 data. Numbers in parentheses are the std. deviations. TS-MKL Linear is the method proposed in Chapter 5. TS-MKL Polynomial-2 and Polynomial-3 learn linear K-classifiers on polynomial K-examples (Eq. 6.1) of degree 2 and degree 3 respectively and thus learn a polynomial kernel combination.

worse than its degree-2 counterpart for datasets. The deterioration in performance can be attributed to overfitting in the K-space due to high dimensionality and limited number of samples.

Polynomial TS-MKL can be expected to give improvements over linear TS-MKL when the set of base kernels is weak as it is the case in the experiments of Table 6.1. In these experiments I used linear kernels on individual features as the base kernels. When the set of best kernels is already rich, Polynomial TS-MKL does not seem to improve performance over linear TS-MKL. This was observed when I used the full set of base kernels for UCI datasets that were used in Chapter 5. Polynomial TS-MKL did not improve accuracy over the numbers reported in Table 5.2. In some cases, it also suffered a loss in performance due to overfitting in K-space.

The same phenomenon was also observed on both Caltech datasets where Polynomial TS-MKL does not improve beyond linear TS-MKL. Results for Caltech-101 are reported below.

6.2.2 Indefinite similarity functions

In the last Section, I experimented with polynomial kernel combinations in the framework of TS-MKL using polynomial K-classifiers that resulted in positive semidefinite kernels. It was observed on Caltech and UCI datasets that when the set of base kernels is rich, the polynomial TS-MKL does not improve over linear TS-MKL. In this Section, I will experiment with other types of nonlinear K-classifiers. In general for most nonlinear K-classifiers $h(\cdot)$, it is not possible to guarantee positive semidefiniteness of the induced similarity function $\tilde{K}_h(\cdot,\cdot) = h(\mathbf{z}_{uv})$. This will preclude it from being used as it is in a kernel based learning method such as SVM. There are two ways to get around this problem. There exist methods that treat the indefinite kernel as a

noisy version of a positive semidefinite kernel and solve a joint objective for the "pure" positive semidefinite kernel and the SVM dual coefficients Luss & dAspremont (2007); Chen & Ye (2008). However, these methods are computationally expensive and so are not suitable for medium to large scale problems.

Another way to use the indefinite kernels in supervised learning can be through the *learning with similarity functions framework* of Balcan & Blum (2006) which seems more appealing due to its applicability to large scale problems. The basic idea in this framework is to select a set of "landmark" points which are not a part of the training data and represent all other points in terms of their similarity with these landmark points. These landmark points consist of equal number of positive and negative class examples. For example, if $\mathbf{t}_1, \ldots, \mathbf{t}_l$ are the l selected landmark points, the feature representation of ith example is given as

$$\phi(\mathbf{x}_i) = [K(\mathbf{x}_i, \mathbf{t}_1),\ K(\mathbf{x}_i, \mathbf{t}_2), \ldots,\ K(\mathbf{x}_i, \mathbf{t}_l)] \in \mathbb{R}^l, \tag{6.3}$$

where $K(\cdot, \cdot)$ is a similarity function. Subsequently, a linear classifier is learned on the examples $\{\phi(\mathbf{x}_i)\}_{i=1}^n$. For a "good" similarity function $K(\cdot, \cdot)$, learning with landmark similarity based representation is guaranteed to have bounded generalization error with polynomial sample complexities (Balcan & Blum, 2006). In Eq. 6.3, the similarity function $K(\cdot, \cdot)$ can be indefinite and so the similarity $\tilde{K}_h(\cdot, \cdot)$ induced by any nonlinear K-classifier $h(\cdot)$ can be used. Moreover, nothing prevents us from using indefinite base kernels since we no longer have positive semidefinite restriction on the final learned kernel.

6.2.2.1 Theoretical perspective

Let us recall the following definition of goodness of a kernel function.

Definition 6.2.1. *A kernel K is an (ϵ, γ)-***good kernel function** *if there exists a vector β, $\|\beta\| \leq 1$ such that*

$$Pr_{(x,y)\sim P}[y\langle\phi(x), \beta\rangle \geq \gamma] \geq 1 - \epsilon$$

A predictor with error at most $\epsilon + \epsilon_{\text{acc}}$ can be learned with high probability from $\tilde{O}((\epsilon + \epsilon_{\text{acc}})/(\gamma^2\epsilon_{\text{acc}}^2))$ examples using an (ϵ, γ)-good kernel function K (Defn. 6.2.1).

Balcan & Blum (2006) define the goodness of a similarity function as follows.

Definition 6.2.2. *A pairwise function K is an (ϵ, γ)-***good similarity function** *for a distribution P if there exists a weighting function $w : X \mapsto [0, 1]$ such that at least a $1 - \epsilon$ fraction of probability mass of examples (x, y) satisfy*

$$E_{(x',y')\sim P}[yy'w(x')K(x, x')] \geq \gamma$$

If we take the weighting function $w(\cdot)$ to be a constant 1 everywhere, this definition says that average similarity between same class examples is 2γ more than the average similarity between different class examples (under balanced class distribution). It was shown by Balcan & Blum (2006) that a predictor with error at most $\epsilon + \epsilon_{\text{acc}}$ can be learned with high probability from $\tilde{O}((\epsilon + \epsilon_{\text{acc}})/(\gamma^2\epsilon_{\text{acc}}^2))$ examples using an (ϵ, γ)-good similarity function K (Defn. 6.2.2). This is achieved by constructing a landmark based representation as outlined in (Balcan & Blum, 2006) and learning a suitable linear predictor in this transformed space.

As mentioned above, kernel-based learning with (ϵ, γ)-good kernel and similarity-based learning with (ϵ, γ)-good similarity function have similar sample complexity guarantees. It is natural to ask which learning framework is better (kernel-based learning or similarity-based learning of Balcan & Blum (2006)) if the given similarity measure is also positive semidefinite. Srebro (2007) proved the following result:

Theorem 6.2.1. *If a kernel K is (ϵ, γ)-good kernel (Defn. 6.2.1) for a consistent input distribution (y is a deterministic function of x), then it is also $(\epsilon + \epsilon_1, 1/2(1-\epsilon)\epsilon_1\gamma^2)$-good similarity function (Defn. 6.2.2) for the distribution (for any $\epsilon > 0$).*

This theorem says that there is a significant degradation in the margin when the kernel is used in similarity based learning framework of Balcan & Blum (2006). Srebro (2007) also proved by giving a counterexample that this result is tight up to a multiplication factor of 16 in the margin.

These results give an indication that although it may be possible to get a better similarity function by sacrificing positive semidefinite property, the gain here can be offset by the use of similarity based representation and one might be better off working with the positive semidefinite kernel based learning. In the following Section, we will test this by doing away with the positive semidefiniteness of learned kernel and will use random forests as our K-classifier. The following Section gives a brief overview of Random Forests for the sake of completeness.

6.2.2.2 Random Forests

Random forest is a collection of tree classifiers where each tree is generated using the same training set but based on a random vector sampled independently from the same distribution for all trees. The final prediction for an input $\mathbf{x}$ is taken to be the class label that is predicted by maximum number of trees in the ensemble. It is defined formally as follows (Breiman, 2001):

Definition 6.2.3. *A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\mathbf{x}, \Theta_k), k = 1, \ldots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$.*

There can be several flavors of random forests based on the randomness injected in the building of trees, as dictated by vectors $\{\Theta_k\}$. I will use the flavor made popular by Breiman (2001) where two sources of randomness are used: *bagging* and *random features*. In bagging, training data for each tree classifier is drawn randomly (with replacement) from the original training set. Bagging improves the performance of unstable predictors (Breiman, 1994). Since the training samples for each tree are drawn randomly with replacement, the out-of-bag (OOB) samples can also be used to get an estimate of the generalization error Breiman (2001). The second source of randomness is the random selection of features at each node in the tree based on which the node is split. It was shown by Breiman (2001) that random forests do not overfit with increasing number of trees and the error converges to a limiting value.

6.2.2.3 Empirical findings

This section describes the experimental methodology and the results obtained on UCI and Caltech datasets. The training data for the random forest are the K-examples constructed same as in linear TS-MKL. Total 500 trees are grown and the minimum size of the terminal nodes is set to 3. For splitting each node in the tree, ten random features are sampled and the feature that gives

	Sonar	Ionosphere	Pima	Vertebral
Average Kernel	74.52(5.9)	87.14(3.4)	75.84(2.5)	81.02(5.3)
TS-MKL Linear	73.10(3.4)	87.00(3.3)	76.43(2.1)	81.13(4.7)
TS-MKL Polynomial-2	83.57(2.9)	90.43(3.1)	76.56(2.8)	82.48(5.3)
TS-MKL Polynomial-3	82.10(5.9)	89.21(3.8)	74.83(2.6)	78.20(4.2)
TS-MKL RF	82.86(4.2)	92.71(3.5)	75.39(2.9)	81.26(6.9)

Table 6.3: Average accuracy (%) over 10 random splits on UCI datasets. Numbers in parentheses are the std. deviations. TS-MKL Linear is the method proposed in Chapter 5. TS-MKL Polynomial-2 and Polynomial-3 learn linear K-classifiers on polynomial K-examples (Eq. 6.1) of degree 2 and degree 3 respectively and thus learn a polynomial kernel combination. TS-MKL RF learn a random forest over linear K-examples, thus learning similarities in a nonlinear fashion.

maximum information gain is selected to split the node. The final similarity value for the example pair is taken to be the ratio of number of trees that predict positive K-label for the corresponding K-example divided by the total number of trees. Similarity values are computed for all pairs of examples in this manner.

The similarities computed above are used in conjunction with learning with similarity functions framework (Balcan & Blum, 2006). I randomly sample equal number of landmark points from each class such that the total number of landmark points is equal to $1/3$th of the total training samples. These landmark points are used to construct the new landmark based representation as in Eq. 6.3. Although Balcan & Blum (2006) assume that landmark points do not belong to the training set for ease of analysis, we include landmark points in the training set due to limited amount of training data available in these datasets. This was also justified theoretically in a recent work (Jain & Kar, 2012). The results are averaged over the five random selection of landmarks. I denote this approach by TS-MKL RF (two stage multiple kernel learning - random forest).

Table 6.3 shows the results for UCI datasets. The set of base kernels is same as that used in the previous Section. I also reproduce the results obtained with polynomial kernel combination for the sake of comparison. The accuracies are slightly worse than polynomial TS-MKL for all datasets except Ionosphere for which TS-MKL RF performs better.

The results for Caltech-101 are shown in Table 6.4. For this data, TS-MKL RF is better than average kernel but is worse than all other methods including linear TS-MKL. Although the 0-1 error in the K-space is less than that for linear TS-MKL, resulting in better similarity, the advantage is offset by the use of learning-with-similarity framework (Srebro, 2007). The loss in performance is more pronounced than UCI datasets which may be because of limited number of samples per class (102 classes and not more than 30 samples per class). In each one-vs-rest subproblem, there are 10 landmark points from $+$ve class and 1010 landmark points from $-ve$ class (for 30 samples per class case), resulting in class imbalance in the similarity based representation. This imbalance cannot be corrected without further loss in performance since correcting it results in too few number of total landmark points (20 from $+$ve class and 20 from $-ve$ class). In these experiments, the learning-with-similarity framework was not found to be performing comparably with traditional learning-with-kernel framework for multi-class problems.

	Training samples per class					
	5	10	15	20	25	30
Average Kernel	44.40 (0.51)	55.69(0.57)	62.35(1.07)	66.22(0.95)	68.96(0.99)	71.68(1.38)
TS-MKL Linear	51.23 (0.51)	62.99(0.86)	69.05(0.88)	72.49(0.27)	75.21(0.66)	77.16(0.88)
TS-MKL Polynomial-2	50.75(0.41)	62.94(0.76)	67.71(0.88)	71.69(0.48)	74.87(0.81)	75.14(0.73)
TS-MKL Polynomial-3	49.36(0.63)	60.29(0.82)	67.23(0.75)	70.09(0.74)	73.61(0.93)	74.31(0.93)
TS-MKL RF	48.18(1.21)	58.74(0.98)	64.18(1.13)	69.03(1.24)	71.63(1.05)	72.23(0.97)

Table 6.4: Average accuracy (%) over 5 random splits on Caltech-101 data. Numbers in parentheses are the std. deviations. TS-MKL Linear is the method proposed in Chapter 5. TS-MKL Polynomial-2 and Polynomial-3 learn linear K-classifiers on polynomial K-examples (Eq. 6.1) of degree 2 and degree 3 respectively and thus learn a polynomial kernel combination. TS-MKL RF learn a random forest over linear K-examples, thus learning similarities in a nonlinear fashion.

6.3 Discussion

The ideas of previous chapter were extended to learn nonlinear kernel combinations in two ways – (i) learning positive definite kernels using polynomial kernel combination (TS-MKL Polynomial), (ii) learning indefinite similarities using random forest (TS-MKL RF). Polynomial TS-MKL of degree-2 improves the performance beyond linear TS-MKL if the set of base kernels is not rich enough, as it was observed in experiments with UCI datasets (Table 6.1). Polynomial TS-MKL of degree-3 tends to be a little worse than degree-2 variant. Indefinite similarities are learned using random forests in K-space and learning-with-similarity-functions (LWS) framework of Balcan & Blum (2006) is used to build classifiers with these similarities. The LWS framework seems to affect the prediction performance adversely and the obtained results are worse than polynomial kernel combination for both UCI and Caltech datasets and worse than linear kernel combination for Caltech datasets. This can be explained in the light of results in Srebro (2007). The hit in performance was more pronounced for multi-class case with limited number of samples per class as in Caltech data.

Chapter 7

Conclusions and Future Work

This thesis deals with the scenario where the data is available in terms of similarities between pairs of examples and there are multiple similarities for each pair of examples. These multiple similarities can originate either from different types of similarity measures defined on the example pairs or from a single similarity measure defined on multiple data representations. The primary contributions of this thesis lie in developing algorithms that can take advantage of this multimodal information in the context of supervised and unsupervised learning. The contributions of this thesis include:

- Developing algorithms for spectral embedding from multiple similarity graphs, using the ideas of co-training (Kumar & Daumé III, 2011) and co-regularization (Kumar et al., 2011).
- Developing a two-stage method for linear multiple kernel learning (TS-MKL) that transforms the problem of kernel learning to that of binary classification in a transformed space (K-space). The ideas of this method were also extended to nonlinear multiple kernel learning, specifically for polynomial kernel combinations resulting in positive semidefinite kernels (TS-MKL Polynomial) and for general nonlinear kernel combinations using random forests resulting in indefinite similarity (TS-MKL RF).

7.1 Future Directions

There are several interesting directions that can be explored in the future, some of which are already mentioned in previous chapters. The other directions include:

- **Extension of co-trained and co-regularized spectral embedding to missing data scenario:** It will be interesting to extend the proposed methods in Chapter 3 and Chapter 4 to the setting where some of the similarities are missing in one of the views. Some recent works have explored this direction (Rai et al., 2010).
- **New quality functions for linear multiple kernel learning:** The problem of linear MKL is transformed into a binary classification problem in another space (K-space) and a novel quality function is proposed for learning kernel weights that minimized hinge loss in the K-space. The resulting approach, linear TS-MKL, performs better or on par with leading MKL methods. It will be useful to investigate new criteria for measuring kernel quality.
- **New methods for nonlinear multiple kernel learning:** I studied two methods for learning nonlinear kernel combinations building on ideas from linear TS-MKL. These methods were

not able to outperform linear TS-MKL on various benchmark datasets. It will be useful to investigate new methods for learning nonlinear kernel combinations and show convincing improvements over linear MKL methods.

Besides above directions, it will also be interesting to use the ideas of Chapter 5 for kernel learning in scarce data conditions and kernel learning for semi-supervised clustering, as discussed in Section 5.4.

Bibliography

Aizerman, M., Braverman, E., and Rozonoer, L. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

Amini, Massih-Reza, Usunier, Nicolas, and Goutte, Cyril. Learning from multiple partially observed views - an application to multilingual text categorization. In *Advances in Neural Information Processing Systems*, 2009.

Aronszajn, N. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, pp. 337–404, 1950.

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

Bach, F., Lanckriet, G. R. G., and Jordan, M. I. Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning*, 2004.

Bach, Francis R and Jordan, Michael I. Learning spectral clustering, with application to speech separation. *The Journal of Machine Learning Research*, 7:1963–2001, 2006.

Bakker, Bart and Heskes, Tom. Task Clustering and Gating for Bayesian Multitask Learning. *Journal of Machine Learning Research*, 4, 2003.

Balcan, Maria-Florina and Blum, Avrim. On a Theory of Learning with Similarity Functions. In *ICML*, 2006.

Balcan, Maria-Florina, Blum, Avrim, and Yang, Ke. Co-training and expansion: Towards bridging theory and practice. In *NIPS*, 2004.

Bartlett, P. and Mendelson, S. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 3, 2002.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, pp. 1373–1396, 2003.

Bickel, S. and Scheffer, T. Multi-View Clustering. In *IEEE International Conference on Data Mining*, 2004.

Biehl, Michael, Hammer, Barbara, Verleysen, Michel, and Villmann, Thomas. *Similarity based clustering*. Springer, 2009.

Blaschko, M. B. and Lampert, C. H. Correlational Spectral Clustering. In *Computer Vision and Pattern Recognition*, 2008.

Blei, David M., Ng, Andreq Y., and Jordan, Michael I. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, pp. 993–1022, 2003.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Conference on Learning Theory*, 1998.

Bo, Liefeng, Ren, Xiaofeng, and Fox, Dieter. Multipath sparse coding using hierarchical matching pursuit. In *NIPS workshop on deep learning*, 2012.

Boser, Bernhard E., Guyon, Isabelle M., and Vapnik, Vladimir N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory (COLT)*, 1992.

Boyd-Graber, Jordan and Blei, David M. Multilingual topic models for unaligned text. In *Uncertainty in Artificial Intelligence (UAI)*, 2009.

Breiman, Leo. Bagging predictors. *Machine Learning*, 1994.

Breiman, Leo. Random forests. *Machine Learning*, 45, 2001.

Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

Callison-Burch, Chris and Osborne, Miles. Co-training for statistical machine translation. In *Proceedings of the 6th Annual CLUK Research Colloquium*, 2003.

Caruana, Rich. Multitask Learning. *Machine Learning*, 28, 1997.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

Chaudhuri, Kamalika, Kakade, Sham M., Livescu, Karen, and Sridharan, Karthik. Multi-view Clustering via Canonical Correlation Analysis. In *International Conference on Machine Learning*, 2009.

Chen, Jianhui and Ye, Jieping. Training svm with indefinite kernels. In *ICML*, 2008.

Chen, Minmin, Weinberger, Kilian Q., and Blitzer, John C. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems*, 2011.

Cortes, C. and Vapnik, V. Support Vector Networks. *Machine Learning*, 20(3), 1995.

Cortes, C., Mohri, M., and Rostamizadeh, A. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems*, 2009a.

Cortes, C., Mohri, M., and Rostamizadeh, A. Generalization bounds for learning kernels. In *International Conference on Machine Learning*, 2010a.

Cortes, C., Mohri, M., and Rostamizadeh, A. Two-Stage Learning Kernel Algorithms. In *International Conference on Machine Learning*, 2010b.

Cortes, Corinna, Mohri, Mehryar, and Rostamizadeh, Afshin. Learning non-linear combination of kernels. In *Advances in Neural Information Processing Systems*, 2009b.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. On Kernel-Target Alignment. In *NIPS*, 2001.

Daumé III, H., Kumar, A., and Saha, A. Co-regularization Based Semi-supervised Domain Adaptation. In *NIPS*, 2010.

de Sa, Virginia R. Spectral Clustering with two views. In *Proceedings of the Workshop on Learning with Multiple Views, International Conference on Machine Learning*, 2005.

Dhillon, Inderjit S., Guan, Yuqiang, and Kulis, Brian. Kernel k-means, spectral clustering and normalized cuts. In *KDD*, 2004.

Ekin, Ahmet, Pankanti, Sharath, and Hampapur, Arun. Initialization-independent spectral clustering with applications to automatic video analysis. In *ICASSP*, volume 3, pp. iii–641, 2004.

Emanuelsson, O., Nielsen, H., Brunak, S., and von Heijne, G. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *Journal of Molecular Biology*, 300: 1005–1016, 2000.

Gardy, J. L., Laird, M. R., Chen, F., Rey, S., Walsh, C. J., Ester, M., and Brinkman, F. S. L. PSORTb v.2.0: expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis. *Bioinfomatics*, 21:617–623, 2004.

Gehler, P. and Nowozin, S. On Feature Combination for Multiclass Object Detection. In *International Conference on Computer Vision*, 2009.

Gehler, P.V. and Nowozin, S. Infinite kernel learning. In *NIPS 2008 Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, 2008.

Gonen, Mehmet and Alpaydin, Ethem. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, pp. 2211–2268, 2011.

Greene, D. and Cunningham, P. Producing accurate interpretable clusters from high-dimensional data. In *PKDD*, 2005.

Greene, Derek and Cunningham, Pádraig. A matrix factorization approach for integrating multiple data views. In *European Conference on Machine learning*, 2009.

Hofmann, Thomas. Probabilistic latent semantic analysis. In *Uncertainty in Artificial Intelligence*, 1999.

Horn, Roger A. and Johnson, Charles R. *Matrix Analysis*. 1990.

Hubert, Lawrence and Arabie, Phipps. Comparing Partitions. *Journal of Classification*, pp. 193–218, 1985.

Jacob, Laurent, Bach, Francis, and Vert, Jean-Philippe. Clustered Multi-task Learning: a Convex Formulation. In *NIPS*, 2008.

Jagarlamudi, Jagadeesh and Daumé III, Hal. Extracting multilingual topics from unaligned corpora. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, 2010.

Jain, Prateek and Kar, Purushottam. Supervised learning with similarity functions. In *NIPS*, 2012.

Joachims, T. Text categorization with support vector machines: learning with many relevant features. In *ECML*, 1998.

Kandola, J. S., Shawe-Taylor, J., and Cristianini, N. Optimizing Kernel Alignment over Combination of Kernels. In *Tech. Report 121, Dept. of CS, Univ. of London, UK*, 2002.

Kar, Purushottam. Generalization guarantees for a binary classification framework for two-stage multiple kernel learning. *arXiv:1302.0406*, 2013.

Kloft, M., Brefeld, U., Sonnenburg, S., and Zien, A. ℓ_p-Norm Multiple Kernel Learning. *Journal of Machine Learning Research*, 12:953–997, 2011.

Kumar, Abhishek and Daumé III, Hal. A Co-training Approach for Multiview Spectral Clustering. In *International Conference on Machine Learning*, 2011.

Kumar, Abhishek, Rai, Piyush, and Daumé III, Hal. Co-regularized multiview spectral clustering. In *NIPS*, 2011.

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

Liberty, Edo, Woolfe, Franco, Martinsson, Per-Gunnar, Rokhlin, Vladimir, and Tygert, Mark. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 2007.

Luss, Ronny and dAspremont, Alexandre. Support vector machine classification with indefinite kernels. In *NIPS*, 2007.

Mangasarian, O. L. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

Manning, Christopher D., Raghavan, Prabhakar, and Schtze, Hinrich. *Introduction to Information Retrieval*. 2008.

Mei, Qiaozhu, Cai, Deng, Zhang, Duo, and Zhai, ChengXiang. Topic modeling with network regularization. In *World Wide Web (WWW)*, 2008.

Mercer, James. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Royal Society London*, 209:415–446, 1909.

Moguerza, Javier M. and Munoz, Alberto. Support vector machines with applications. *Statistical Science*, 21:322–336, 2006.

Mostafavi, Sara and Morris, Quaid. Fast integration of heterogeneous data sources for predicting gene function with limited annotation. *Bioinformatics*, pp. 1759–1765, 2010.

Müller, Christoph, Rapp, Stefan, and Strube, Michael. Applying co-training to reference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 2002.

Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems*, 2002.

Ni, Xiaochuan, Sun, Jian-Tao, Hu, Jian, and Chen, Zheng. Mining multilingual topics from wikipedia. In *World Wide Web (WWW)*, 2009.

Nigam, Kamal and Ghani, Rayid. Analyzing the Effectiveness and Applicability of Co-training. In *International Conference on Information and Knowledge Management*, 2000.

Ning, Huazhong, Xu, Wei, Chi, Yun, Gong, Yihong, and Huang, Thomas S. Incremental spectral clustering with application to monitoring of evolving blog communities. In *SDM*, 2007.

Niu, Donglin, Dy, Jennifer G., and Jordan, Michael I. Multiple non-redundant spectral clustering views. In *International Conference on Machine Learning*, 2010.

Ong, C. S., Smola, A., and Williamson, R. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

Orabona, F. and Jie, L. Ultra-fast optimization algorithm for sparse multi kernel learning. In *International Conference on Machine Learning (ICML)*, pp. 249–256, 2011.

Poggio, T. On optimal nonlinear associative recall. *Biological Cybernetics*, 19:201–209, 1975.

Rai, Piyush, Trivedi, Anusua, Daumé III, Hal, and DuVall, Scott L. Multiview clustering with incomplete views. In *NIPS 2010: Workshop on Machine Learning for Social Computing*, 2010.

Rakotomamonjy, A., Bach, F., Canu, S., and Grandvalet, Y. More efficiency in multiple kernel learning. In *International Conference on Machine Learning*, 2007.

Sarkar, Anoop. Applying co-training methods to statistical parsing. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, 2001.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *International Conference on Machine Learning*, 2007.

Sindhwani, V. and Lozano, A. C. Non-parametric group orthogonal matching pursuit for sparse learning with multiple kernels. In *NIPS*, pp. 2519–2527, 2011.

Sindhwani, Vikas, Niyogi, Partha, and Belkin, Mikhail. A Co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of the Workshop on Learning with Multiple Views, International Conference on Machine Learning*, 2005.

Smith, F. W. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, pp. 367–372, 1968.

Smola, Alexander J., Bartlett, Peter, Schlkopf, Bernhard, and (Eds.), Dale Schuurmans. Advances in large margin classifiers, 1999.

Smola, B. Scholkopf A. and Muller, K.-R. Kernel Principal Component Analysis. *Advances in Kernel Methods - Support Vector Learning*, pp. 327–352, 1999.

Sonnenburg, S., Ratsch, G., Schafer, C., and Scholkopf, B. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7, 2006.

Srebro, N. How Good is a Kernel When Used as a Similarity Measure. In *COLT*, 2007.

Strehl, Alexander and Ghosh, Joydeep. Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, pp. 583–617, 2002.

Tang, Wei, Lu, Zhengdong, and Dhillon, Inderjit S. Clustering with Multiple Graphs. In *IEEE International Conference on Data Mining*, 2009.

Tay, Francis E.H and Cao, Lijuan. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.

Tsang, Ivor Wai-Hung and Kwok, James Tin-Yau. Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks*, 17:48–58, 2006.

UCI. The UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/.

Valgren, Christoffer, Duckett, Tom, and Lilienthal, Achim. Incremental spectral clustering and its application to topological mapping. In *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4283–4288, 2007.

Vapnik, V. *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow, 1979.

Vapnik, V. and Chervonenkis, A. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

Vapnik, V. and Lerner, A. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

Varma, Manik and Babu, Bodla Rakesh. More generality in efficient multiple kernel learning. In *International Conference on Machine Learning*, 2009.

von Luxburg, Ulrike. A Tutorial on Spectral Clustering. *Statistics and Computing*, 2007.

Wang, Xiang, Qian, Buyue, and Davidson, Ian. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, pp. 1–30, 2012.

Xiao, Jianxiong and Quan, Long. Multiple view semantic segmentation for street view images. In *ICCV*, 2009.

Xue, Ya, Liao, Xuejun, Carin, Lawrence, and Krishnapuram, Balaji. Multi-Task Learning for Classification with Dirichlet Process Priors. *Journal of Machine Learning Research*, 8, 2007.

Yang, Z. R. Biological applications of support vector machines. *Brief. Bioinform.*, 2004.

Yi, Xing, Xu, Yunpeng, and Zhang, Changshui. Multi-view em algorithm for finite mixture models. In *ICAPR, Lecture Notes in Computer Science, Springer-Verlag*, 2005.

Zhao, Bin, Kwok, James T., and Zhang, Changshui. Multiple Kernel Clustering. In *SIAM International Conference on Data Mining*, 2009.

Zhou, Dengyong and Burges, Christopher J. C. Spectral Clustering and Transductive Learning with Multiple Views. In *International Conference on Machine Learning*, 2007.

Zien, A. and Ong, C. S. Multiclass Multiple Kernel Learning. In *International Conference on Machine Learning*, 2007.