NOVEL ALGORITHMS FOR LARGE SCALE SUPERVISED AND ONE CLASS LEARNING

by

Lingyan Sheng

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

May 2013

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Prof. Antonio Ortega for his continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm, and knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. C.-C. Jay Kuo, Prof. Yan Liu, as well as my qualify exam committee: Prof. B. Keith Jenkins and Prof. Fei Sha for their encouragement, insightful comments, and hard questions.

My sincere thanks also goes to Prof. Shahab Asgharzadeh and Dr. Roger Pique-Regi, for their essential support and help through my project with Children's Hospital Los Angeles.

I thank my fellow labmates in the Compression Group for the stimulating discussions.

Last but not the least, I would like to thank my family: my parents Zhiren Sheng and Yaqin Wu, my husband Tao Chen and my daughter Eileen Chen. Without their encouragement and understanding it would have been impossible for me to finish this work.

Table of Contents

List of Tables

List of Figures

Abstract

Supervised learning is the machine learning task of inferring a function from labeled training data. There have been numerous algorithms proposed for supervised learning, such as linear discriminant analysis (LDA), support vector machine (SVM), decision trees, and etc. However, most of them are not able to handle an increasingly popular type of data, high dimensional data, such as gene expression data, text documents, MRI images, and etc. This phenomenon is often called the curse of dimensionality. Our solution to this problem is an improvement to LDA that imposes a regularized structure on the covariance matrix, so that it becomes block diagonal while feature reduction is performed. The improved method, which we call block diagonal discriminant analysis (BDLDA), effectively exploits the off diagonal information in the covariance matrix without huge computation and memory requirement. BDLDA is further improved by using treelets as a preprocessing tool. Treelets, by transforming the original data by successive local PCA, concentrates more energy near the diagonal items in the covariance matrix, and thus achieves even better accuracy compared to BDLDA.

Supervised learning requires labeled information of all classes. However, since labeled data is often more difficult to obtain than unlabeled data, there is an increasing interest in a special form of learning, namely, one class learning. In one class learning, the training set only has samples of one class, and the goal is to

distinguish the class from all other samples. We propose a one class learning algorithm, Graph-One Class Learning (Graph-OCL). Graph-OCL is a two step strategy, where we first identify reliable negative samples, and then we classify the samples based on labeled data and the identified negative samples in the first step. The main novelty is the first step, in which graph-based ranking by learning with local and global consistency (LGC) is used. Graph-based ranking is particularly accurate if the samples and their similarities are well represented by a graph. We also theoretically prove that there is a simple method to select a constant parameter α for LGC, thus eliminating the necessity of model selection by time consuming validation.

Graph-based methods usually scale badly as a function of the sample size. This can be solved by using the Nyström approximation, which samples a few columns to represent the affinity matrix. We propose a new method, BoostNyström, which adaptively samples a subset of columns at each iterative step and updates the sampling probability in the next iterative step. This algorithm is based on a novel perspective, which relates the quality of Nyström approximation with the subspace spanned by the sampled columns. BoostNyström can be potentially applied to Graph-OCL to solve the problem of large data size.

Chapter 1

Introduction

This thesis focuses on two machine learning problems: supervised learning and one class learning. They are both classification problems, but they differ in one important aspect, namely how the training samples are labeled. A supervised learning algorithm analyzes the training data, which includes labeled data in all classes, and produces an inferred classification function. The classification function is then applied to the test data to find their classes. Supervised learning finds applications in many real life problems. One example that many people might have encountered is the classification of spam emails and normal emails. Supervised learning is also used in many academic fields other than computer science. For example, it is used to classify the gene expression data, which can help medical doctors to predict the survival rate and identify possibly responsible genes for certain diseases. We will discuss this application in more detail in Chapter 2.

One class learning, on the other hand, has labeled data that belongs to only one of the classes. Only unlabeled data is available in all the other classes. There are two reasons for the increasing popularity of one class learning. One reason is that usually labeled data is more difficult to obtain than unlabeled data. Labeling data may require special skills and equipment, which can be expensive to obtain. [9] described

one situation that only one class has labeled data. In industrial production, there is usually enough data available to characterize the state of the system when it is operating correctly. However, there might not be enough information about how the system malfunctions, because it certainly is not a good idea to disrupt the system in order to obtain the corresponding negative data. If we want to classify future samples, we have to build a model based on one class learning. The other motivation for one class learning is that only one class has characteristic samples, and other classes might not have typical samples. For instance, an e-business website wants to classify the customers into valued customers and normal customers. They may have well defined characteristics for valued customer, such as high shopping frequency, large number of purchases or a large amount of money spent in certain period of time. On the other hand, normal customers do not have typical characteristics to describe their shopping behavior. Thus, we may need to do a one class learning in which only the class of valued customers has labeled data.

There have been various algorithms proposed to solve supervised learning and one class learning problems. Many times, the one class learning algorithm is derived from a supervised learning algorithm. For example, [37] designed an adaptive SVM method for one class learning of text data, and SVM is traditionally used in supervised learning. Similarly, [33] extended the naive bayesian method, a traditional supervised learning method, to one class learning. With the advent of large data in different fields, there emerge new challenges in applying the algorithms. These challenges can be due to either high dimension or large sample size, that often arises in big data problems. Each situation leads to challenges in many traditional algorithms. This thesis will focus on large data challenges in supervised learning and one class learning. We will first discuss a challenge in supervised learning, which

analyzes high dimensional data sets. Then we discuss a novel one class learning algorithm, and the challenge of scaling it up to large sample size.

1.1 Supervised Learning of High Dimensional Data

The goal of supervised learning is to build a model that captures the intrinsic associations between the class type and the features, so the class associated to a sample can be accurately predicted from its feature values. For this purpose, the data is usually divided into a training set and a test set, where the training set is used to build the classifier which is validated by the test set. There are several models developed for supervised learning, e.g., Naive Bayesian, neural networks, decision trees [36], SVM [53], LDA [42] [21], and so on.

One of the challenges of supervised learning is to build a classification model on high dimensional data sets. The emergence of various new application domains, such as bioinformatics and e-commerce, underscores the need for analyzing high dimensional data. For example, in a gene expression microarray data set, there could be thousands or even millions of dimensions, each of which corresponds to a specific segment of genome.

[54] described two challenges for analyzing high dimensional data. The first one is the curse of dimensionality. Many machine learning algorithms have complexity that is at least quadratic or even exponential with respect to the number of dimensions. With a fixed number of training samples, the predictive power of a learning algorithm decreases as the dimensionality increases. Sometimes, even the prediction is impractical. Secondly, when a measure such as a Euclidean distance

is defined on high dimension, there is little difference in the distances between different pairs of samples [6]. This phenomenon may cause many machine learning algorithms to be vulnerable to the presence of noise.

Among the state-of-art supervised learning algorithms, SVM is one of the most successful classification models. An SVM model represents the samples as points in space, and derives a hyperplane as a boundary between separate categories to make the margin as wide as possible. SVM achieves the nonlinear classification by using the kernel trick, which transforms the input space into another higher dimensional feature space. Without having to express the data explicitly in higher space, only the kernel matrix is necessary to obtain the solution. SVM is a natural fit for high dimensional data, because through the dual problem in convex optimization, its complexity only depends on the sample size. However, the complexity of training an SVM is $O(N^2)$ where N is the number of samples in the training data set. This could be a big disadvantage when the training data set is large. The other disadvantage is the difficulty to identify the top features that are responsible for the classification if the data is transformed to higher dimensions, since only the kernel matrix is involved in the solution.

Unlike SVM, LDA is a generative learning algorithm, which assumes that samples of a class are linearly separable from samples of other classes, and each of them has a conditional probability distribution that is normally distributed with common covariance matrix. Normal distribution is not a necessary condition to apply the LDA model. Actually many real life data sets do not have a strictly normal distribution. However, normal distribution guarantees optimality of the LDA solution. LDA has simple closed form solution and is effective in many applications [21]. However given insufficient sample size, LDA becomes impractical because of the singularity of the estimated covariance matrix. A few modifications

to solve the overfitting problem have been proposed. One solution is to regularize the covariance matrix by imposing a diagonal structure on it. Examples of taking the assumption of diagonal covariance matrix include [13] [39]. There are also different ways of selecting the features involved in the classification. The features can be selected in a batch mode, or selected sequentially by a certain criterion. The regularization form and how the features are selected are influential factors to decide the classification accuracy. Chapter 2 will focus on our novel method for the regularization form and feature selection.

We have proposed and improved a novel supervised learning method, block diagonal linear discriminant analysis (BDLDA), for high dimensional data, especially gene expression data. BDLDA is based on LDA, but does greedy feature selection in order to maintain the most informative elements in the covariance matrix. Learning all elements in the full covariance matrix is impossible with limited sample size. On the other hand, much useful information is lost if a diagonal covariance matrix is assumed. BDLDA achieves a tradeoff between a full covariance matrix and a diagonal one. The features are selected based on a simple metric, the discriminant function, such that the most discriminant features found are added into the feature set.

Though BDLDA is more accurate as compared with some state-of-art algorithms, there is still much information in the covariance matrix not considered in building the model. One solution is to increase the size and the number of blocks used in BDLDA, but this comes at the cost of increasing complexity. We propose to use treelets as a preprocessing feature reduction method. Treelets provides a tool for feature transformation that imitates successive PCA. It localizes the energy of the covariance matrix near the diagonal. As a preprocessing method, it allows BDLDA to extract more informative features while maintaining the same complexity with

regular BDLDA, and thus leads to better classification accuracy. The classifier combining BDLDA and treelets is called Treelets-BDLDA (T-BDLDA).

The details of the algorithm and experimental results are described in Chapter 2.

1.2 One Class Learning and Extension to Large Data

Supervised learning tries to distinguish between two or more classes with the training set containing samples from all the classes. But as already mentioned, usually labeled data is more difficult to obtain than unlabeled data. This leads to the situation where labeled data may be missing in certain classes. Therefore, nowadays, people are becoming more interested in one special case of learning: one class learning. There is one other term, outlier detection, which is often used for this problem. The term originates from a different application of one class learning, in which the objective is to identify as many negative samples as possible.

The one class classification problem is different in one important aspect from supervised learning. In one class classification only one of the classes has labeled information that can be used in the training step. The other classes only have unlabeled data. We name the class with labeled data as positive class, and all other classes as negative class. The task is to learn a model that defines a boundary between the positive and negative class such that classification accuracy is optimized. The task is similar to semi-supervised learning in the sense that both labeled and unlabeled data are used in the training step to build a model. Unlike in semi-supervised learning however, the boundary is very likely to be biased, because generally the amount of unlabeled data is much larger than that of labeled data.

In One class learning one encounters the same problem as in supervised learning, in that the labeled set of samples may not characteristic enough. Limited number of labeled samples or noise in the measurements may cause the sampling distribution deviate significantly from the real distribution. For example, the labeled data may be concentrated near the boundary, which will complicate the learning process. This problem may be more prominent in one class learning, because we have even less labeled information than in supervised learning. Though the problem is important, we will not discuss it in this thesis. We assume that there is enough training data to determine the characteristics of the labeled class.

Several algorithms have been proposed for one class learning. Koch [26] used neural network for the detection of positive samples, but the resultign algorithm does not have a good performance in case of high dimensions. There have been algorithms proposed that are based on the Bayesian approach, such as Naive Bayesian [35] and S-EM [33]. Naive Bayesian is simple to implement, but it requires knowledge of prior distributions of the parameters and is not very accurate in deriving the classification boundary, as shown in Chapter 3. S-EM is an advanced form of Bayesian method, and uses the EM algorithm to predict the classification parameters, but because EM algorithm is an iterative method, S-EM is computationally complex which restricts its wide application. An adaptive SVM algorithm for one class learning was proposed in [37]. The algorithm iteratively refines the SVM boundary until convergence. It has high accuracy, but is also computationally expensive and cannot scale up to high sample size.

Recently, there has been a lot of research on spectral graph methods applied to clustering and semi-supervised learning, such as Mincut [7], graph random walk [51], Gaussian Random Fields [63], LGC [58], spectral graph transducer [24], manifold

regularization [5] [49], and many other variants. In particular, [5] generalizes graph-based learning to the manifold setting. In these graph based algorithms, samples are represented as graph nodes, and their similarities are represented by edge weights. Different algorithms may define different regularization forms on the graph. The common goal is to infer a classification function that conforms to the initial labeling and is also smooth on the graph. Graph based methods are generally considered to have high accuracy and low complexity.

To the best of our knowledge, graph based learning algorithms have never been applied to one class learning. Due to their success in semi-supervised learning, graph based algorithms are well suited for learning tasks including both labeled and unlabeled data. However, how graph based algorithms can handle missing labeled information in negative class is not obvious. We propose a novel one class learning algorithm, Graph-OCL, which uses a graph based method, LGC, combined with Transductive-SVM (TSVM). One class learning is usually solved in a two step strategy. The first step is to detect reliable negative samples, which are the most distant samples from the labeled ones. In our proposed algorithm, the reliable negative samples are selected as the lowest ranked samples by LGC ranking. This procedure can be explained by the class probability propagation among the graph, so that the negative samples have the lowest probability belonging to the labeled class. Since graph based methods explore the manifold structure of the data, LGC ranking provides more accurate identification than other methods, such as Naive Bayesian and Rocchio classifier. In order to decide the number of negative samples selected, we use the spy sample technique first proposed in [33]. A few spy samples from the positive labeled set are placed into the unlabeled set. The spy samples provide a calibration so that the samples with the classification probability significantly higher than spies are considered reliable negative.

LGC has one parameter α that controls the balance between graph energy and labeled data fit. Usually the parameters of a learning algorithm are selected by validation methods, such as cross validation. However, it can be proved that LGC ranking is stable when α is smaller than a bound, which is determined by the similarity matrix of the graph. We also show by extensive experiments that LGC has higher accuracy when α is below the bound than when it is above the bound. Thus, α can be chosen as any value below the bound.

The biggest problem with graph based algorithms is that they are unable to scale up to large sample size, although they are effective in small size data. However, for large size problems (e.g. data size>10,000) both storing the similarity matrix, which is a Gram matrix, and solving the associated machine learning problem are prohibitive. This limits usage of the graph-based algorithms in many applications involving large data size, such as sensor networks, social networks, Internet text, etc. In many cases the Gram matrix can be approximated by a sparse matrix, and the sparse matrix computation algorithms can be used. In many other cases the Gram matrix may be dense. In this case, may calculations such as the matrix inversion in LDA, the quadratic programming in SVM, and the computation of the eigendecomposition will take space complexity $O(n^2)$ and time complexity $O(n^3)$. Thus, complexity may be prohibitive when the number of samples n is large.

Recently, many methods have been proposed to approximate the Gram matrix using a low rank structure. For example, [1] used randomized methods to speed up kernel PCA by replacing the kernel matrix (a Gram matrix) by a randomized kernel which approximates the original one in expectation. [55] proposed to use uniform sampling without replacement to choose a small set of columns, from which an approximation to the Gram matrix is built. This method is the so called Nyström approximation method, which derives its name from the Nyström method in integral

equation theory. There have been various random sampling methods proposed for Nyström approximation methods other than uniform sampling, such as [50], [17], [55], [2].

All Nyström approximation methods based on random sampling have one common problem as they may lead to high variance in the approximation errors. Different sampling may lead to different approximation error, some of which may be large. Since in some applications the approximation is run only once, there is no guarantee that a good approximation can be achieved. To solve this problem, we propose an iterative algorithm, BoostNyström. In each iteration, a small number of columns are added into the column set. The added columns have high accuracy, which guarantees a good approximation at each step. The sampled columns in the current iteration are also used to update the sampling probability distribution in the next iteration. BoostNyström has the advantage of reducing the variance in approximation performances by dividing the sampling into smaller subsets of columns that have low approximation error.

BoostNystöm is based on a novel perspective on the Nyström approximation, which states that a good Nyström approximation can be achieved when the space spanned by the sampled columns has a large overlap with the ideal space, where the ideal space is the one spanned by the eigenvectors of top eigenvalues of the Gram matrix. This property guarantees that the columns added in each iterative step which have good approximation constitute a column set with good approximation power. The algorithm to a large extent reduces the randomness often seen in other random sampling Nyström methods, and provides a stable and accurate approximation.

The details of Graph-OCL and experimental results are described in Chapter 3. The details of BoostNyström and experimental results are described in Chapter 4.

Chapter 2

Supervised Learning of High Dimensional Data

2.1 Introduction

In this chapter, supervised learning on high dimensional data is discussed. Nowadays, there are many application domains where the data is of considerable dimensions, such as geographic information system, computer vision, text documents, etc. For example, in text document classification, each word corresponds to one feature. A long text document can contain tens of thousands of different words, which compose a long feature vector.

In this chapter, we will focus on one example of high dimensional data, namely, DNA microarray data. DNA microarray measures mRNA levels of many genes in the cells or tissues at once. The principle behind microarrays is hybridization between two DNA strands. Single strands of complementary DNA for the genes of interest are attached to a solid surface and arranged in an array. The mRNA of a sample of interest, e.g. a tumor biospsy, is extracted and hybridized to the array. The intensity value at each spot in the array is correlated to the abundance of its RNA transcript in the sample. DNA microarrays are used to measure changes in expression levels, or to genotype or targeted resequencing. The output of a biological

experiment using microarray is an intensity image, which represents the RNA transcript values of genes of interest. Usually each spot on the solid space corresponds to many pixels in the intensity image. In order to obtain a single intensity value for each spot, the corresponding pixels need to be first identified, which is called segmentation, and then summarized, which is called quantification. These steps are important in the process of microarray experiments and analysis. However, they are out of the scope of this thesis. We will only discuss the algorithms applied to the microarray gene expression data after segmentation and quantification.

DNA microarray technology allows researchers to analyze patterns of gene expression simultaneously recorded in a single experiment. Gene expression analysis is important in many medical applications. For example, it is generally believed that gene mutations can lead to cancer. Different gene expression patterns among patients or different tissues can be used for diagnosis or prognosis in cancer research. These data sets have a large number of gene expression values per sample (several thousands to tens of thousands, even millions), and a relatively small number of samples (a few dozens), especially in the case of rare diseases, for which gene expression data for only a few patients is available. This situation requires improvements to traditional learning algorithms.

Traditional linear discriminant analysis (LDA) [21,42] cannot be applied to gene expression data, because of the singularity of the within-class scatter matrix due to the small sample size. Thus, for these data sets some form of feature selection will always be needed. A number of solutions based on LDA have been proposed to tackle this challenge. One solution is to assume a diagonal covariance matrix, which essentially ignores potential correlation between different features. Examples include diagonal linear discriminant analysis (DLDA) [13] or nearest shrunken centroid (NSC) [52], as well as sequential DLDA (SeqDLDA) [39], a modified DLDA

technique that incorporates embedded feature selection. Alternative solutions use regularization methods to impose a structure on the covariance matrix, e.g., in shrunken centroid regularized discriminant analysis (SCRDA) [19], a diagonal regularization matrix is employed. But SCRDA has the same problem as DLDA in that it does not perform well in data with correlations (as will be illustrated by our experiments). While it would be possible to consider more complex classification tools (e.g., SVM [53], neural networks [41] and random forests [12]), these tend to not perform as well as simpler LDA-based approaches, e.g., SCRDA, when applied to gene expression data [19]. One likely reason is that these more complex models cannot be accurately learned from limited data. Thus we have a bias-variance trade-off problem: lower variance seen in simple classification methods compensates for the additional bias they introduce [21].

In order to improve performance in the presence of feature correlation (while staying within the general LDA framework), we focus on block diagonal linear discriminant analysis (BDLDA), first proposed in [38]. BDLDA restricts the maximum number of features to be selected in the model. However, even with limited number of features, reliably estimating all correlations is difficult with small sample size. In order to reduce the parameters to be estimated while keeping important correlations between features, BDLDA imposes a block diagonal structure on the covariance matrix. A greedy algorithm is applied to find features to add into candidate models with different block diagonal structures. Cross validation is used to select the best model among all candidate models. Unlike DLDA or NSC, BDLDA performs classification with embedded feature selection, while considering correlations between features. In [38], BDLDA was shown to outperform DLDA on simulated data with sparse covariance structure (e.g., Toeplitz or block diagonal).

While these results were promising, model selection using cross-validation made it impractical for large datasets, e.g., gene expression data.

In this thesis, we improve feature selection in BDLDA by using an estimated error rate to select the best model among all candidate models. The estimated error rate is derived from LDA and can be obtained for each candidate block diagonal covariance structure. Within BDLDA, direct computation of these error rates is possible even when using a very small number of training samples, because the block diagonal structure is limited to use only small blocks. This error rate metric allows us to avoid cross validation for model selection, and enables BDLDA to be computationally practical even when working on large data sets. We apply BDLDA to real gene expression data for the first time, with very competitive results [48]. Other improvements with respect to the original BDLDA approach include a repeated feature subset selection (RFSS) technique and a prescreening procedure. With RFSS, that is repeating model construction with previously selected features removed, the algorithm chooses more discriminating features that are independent from previous models. This is useful for gene expression data, because genes belonging to the same pathway tend to have sparse correlations. The prescreening procedure eliminates features that are not significantly different between two classes, which accelerates model search and improves performance by removing noise. In Section 2.3, test results are presented, that show our improved version of BDLDA works particularly well on simulated data with correlated features and outperforms the other three algorithms in real data.

Though BDLDA has high classification accuracy as compared to some state-of-art algorithms, it still can neglect a significant number of correlations in the covariance matrix, because the assumed block is small. Taking into consideration

both the algorithm complexity and feature correlations, we assume that the maximum block size is 3, because increasing the block size increases BDLDA complexity. A better solution is to transform the covariance matrix so that more of its energy is concentrated on its diagonal elements. For this purpose, we first need to identify the correlated features and perform some form of transformation to decorrelate them. Decorrelating makes it unnecessary to use larger blocks, but we cannot decorrelate globally, so the idea is to perform some local decorrelation, so that smaller blocks can be used.

A popular example of feature transformation and reduction is principal component analysis (PCA) [25]. While PCA can exploit correlation between features to achieve a more compact representation of the feature vectors (thus providing a tool for feature selection), the resulting transformation is done independently of the classification task. Thus, there is no guarantee that projecting samples onto the highest magnitude principal components will lead to better classification performance. Moreover, PCA is a *global* feature transformation (each projection may be obtained as a linear combination of all components of a feature vector). Thus, if a classifier is designed based on PCA data, it will no longer have an easy interpretation in terms of a sparse set of genes, as would be desirable. Other approaches, e.g., those based on wavelets, have been proposed to reduce the impact of noise on classification and, unlike PCA, provide a local analysis [34]. However, wavelets are unlikely to be useful for classification in a DNA context, where there is no reason to expect that adjacent genes would need to have similar levels of expression.

As an alternative, we propose to use treelets as a preprocessing feature reduction tool. Treelets [31] have been proposed as a new adaptive multiscale basis for sparse unordered data. Rather than performing PCA on the whole data (which, for small training set means that only a few global eigenvalues will be identified), in

treelets a tree based local (pair-wise) PCA is applied. Based on the correlation between pairs of features a series of Jacobi rotations [18] are selected and successively applied. Each step in this process involves two features that have high correlation and are diagonalized. This leads to a covariance matrix in which more energy is concentrated on the diagonal than in the original covariance matrix. Thus, treelets can be seen as a feature transformation tool that approximates the decorrelating properties of PCA, but where tree-structured pair-wise processing makes it possible to work with small sample sets.

A key advantage of treelets over PCA is that the feature transformation they produce is local in nature. That is, one can identify (by tracing back the successive pair-wise PCA operations) a sparse set of features (at most 2^l features in a depth l tree) that were transformed to obtain a given treelet coefficient. However, as for PCA, high energy treelet coefficients (analogous to eigenvectors corresponding to high magnitude eigenvalues) are not necessarily guaranteed to provide the best classification performance. Thus, we propose to use treelet coefficients generated from the data as an input to BDLDA [47].

A first advantage of applying BDLDA is that it can take advantage of the energy compaction/denoising properties of treelets, so that feature selection in BDLDA is more reliable. Second, and more important, since each treelet coefficient is obtained by transforming a set of features in the original representation, the blocks are effectively larger, since they operate on transformed features. This allows us to take into consideration the correlation across a larger number of features, without increasing significantly the BDLDA complexity. For publicly available two-class cancer data, T-BDLDA outperforms state of the art techniques, including BDLDA [48], SeqDLDA [39], NSC [52] and SCRDA [19].

This chapter is organized as follows. Section 2.2 describes our proposed BDLDA algorithm. Section 2.2.4 describes the treelets algorithm and its properties. Section 2.2.5 introduces the algorithm combining BDLDA and treelets. Section 2.3 provides the experimental results of both BDLDA and TBDLDA. Section 2.4 concludes the chapter.

2.2 BDLDA Algorithm Description

2.2.1 Model Selection Metric

We start by deriving the estimated error rate of LDA, which will be used as a model selection metric in Section 2.2.2. LDA assumes that both class A and class B have multivariate Gaussian distribution with means $\boldsymbol{m_A}$ and $\boldsymbol{m_B}$ and a common covariance matrix $\boldsymbol{K}$, $f_A(\boldsymbol{x}) \sim N(\boldsymbol{m_A}, \boldsymbol{K})$, $f_B(\boldsymbol{x}) \sim N(\boldsymbol{m_B}, \boldsymbol{K})$. The discriminant function is

$$g(\boldsymbol{x}) = \boldsymbol{w^t x} - b \begin{cases} \geq 0 \Rightarrow \text{class A} \\ < 0 \Rightarrow \text{class B} \end{cases} \tag{2.1}$$

where $\boldsymbol{x}$ is the feature vector of the sample to classify, $\boldsymbol{w}$ is a vector orthogonal to the decision hyperplane, and b defines the decision boundary $g(\boldsymbol{x}) = 0$.

The optimal $\mathbf{w}$ lies in the direction that maximizes the variance between/within ratio $J_{\boldsymbol{K}}(\boldsymbol{w})$, where

$$J_{\boldsymbol{K}}(\boldsymbol{w}) = \frac{(\boldsymbol{d^t w})^2}{\boldsymbol{w^t K w}} \tag{2.2}$$

The solution to the optimization is then:

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} J_{\boldsymbol{K}}(\boldsymbol{w}) = \boldsymbol{K}^{-1}\boldsymbol{d} \qquad \boldsymbol{d} = \boldsymbol{m_A} - \boldsymbol{m_B} \tag{2.3}$$

$$\text{with} \qquad b = \frac{\log \pi_B}{\log \pi_A} \tag{2.4}$$

In practice, the mean vectors $\boldsymbol{m_A}$, $\boldsymbol{m_B}$, the covariance matrix $\boldsymbol{K}$, and the prior class probabilities π_A, π_B are usually replaced by the maximum likelihood estimators $\hat{\boldsymbol{m}}_{\boldsymbol{A}}$, $\hat{\boldsymbol{m}}_{\boldsymbol{B}}$, $\hat{\boldsymbol{K}}$, $\hat{\pi}_A$ and $\hat{\pi}_B$. The discriminant function then becomes:

$$J_{\hat{\boldsymbol{K}}}(\boldsymbol{w}) = \frac{(\hat{\boldsymbol{d}}^{\boldsymbol{t}}\boldsymbol{w})^2}{\boldsymbol{w}^{\boldsymbol{t}}\hat{\boldsymbol{K}}\boldsymbol{w}} \tag{2.5}$$

with a solution:

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} J_{\hat{\boldsymbol{K}}}(\boldsymbol{w}) = \hat{\boldsymbol{K}}^{-1}\hat{\boldsymbol{d}} \qquad \hat{\boldsymbol{d}} = \hat{\boldsymbol{m}}_{\boldsymbol{A}} - \hat{\boldsymbol{m}}_{\boldsymbol{B}} \tag{2.6}$$

$$\text{with} \qquad b = \frac{\log \hat{\pi}_B}{\log \hat{\pi}_A} \tag{2.7}$$

In the general LDA case, if the data set has more features than samples, $\hat{\boldsymbol{K}}$ is not invertible. In BDLDA, we restrict the feature size to be smaller than the sample size, in order to make $\hat{\boldsymbol{K}}$ more likely to be invertible. Different models in BDLDA represent the different size and structure of $\hat{\boldsymbol{K}}$ and $\hat{\boldsymbol{d}}$.

Given the training data, from which the estimated means and covariance matrix are derived, the estimated probability of error [21] of a model in BDLDA is

$$\begin{aligned}\hat{P}_e|T &= \hat{\pi}_A\phi(-\frac{\frac{1}{2}\hat{\boldsymbol{d}}^t\hat{\boldsymbol{K}}^{-1}\hat{\boldsymbol{d}}+\ln(\frac{\hat{\pi}_A}{\hat{\pi}_B})}{\sqrt{\hat{\boldsymbol{d}}^t\hat{\boldsymbol{K}}^{-1}\hat{\boldsymbol{d}}}}) \\ &\quad +\hat{\pi}_B\phi(-\frac{\frac{1}{2}\hat{\boldsymbol{d}}^t\hat{\boldsymbol{K}}^{-1}\hat{\boldsymbol{d}}-\ln(\frac{\hat{\pi}_A}{\hat{\pi}_B})}{\sqrt{\hat{\boldsymbol{d}}^t\hat{\boldsymbol{K}}^{-1}\hat{\boldsymbol{d}}}}) \end{aligned} \tag{2.8}$$

where ϕ is the cdf of the standard normal distribution. T denotes the training data set. Both (2.5) and (2.8) are used as criteria in feature and model selection.

2.2.2 Model Construction and Feature Selection

There are numerous selections of features and covariance structures, such as a full covariance matrix, a diagonal covariance matrix with certain selection of features, or a block diagonal covariance matrix. Enumerating models with all possible selections of features and structures of $\hat{\boldsymbol{K}}$ and $\hat{\boldsymbol{d}}$ is obviously impractical due to computation and memory limitation. In [38], a block diagonal structure is imposed on the covariance matrix, with the dimensions of both subblocks and the resulting covariance matrix kept small. An example of the resulting candidate models is shown in Figure 2.1, where each arrow denotes adding a feature to the model and forming a new model. A feature can be used to start a new subblock (solid line in Figure 2.1) or it can be combined with the current subblock (dashed line in Figure 2.1). The feature that generates the largest $J_{\hat{\boldsymbol{K}}}(\boldsymbol{w})$ in (2.5) among all candidate features is selected. For simplicity, it is assumed that the sizes of subblocks are nonincreasing, e.g., in Figure 2.1, the block size on the top left of the covariance matrix is larger than that on the bottom right. This is based on the assumption that through exhaustive search for features, the features added first are considered of better classification power than those selected later, and thus their correlations

are considered more important. As a consequence, if small blocks are selected at the start of the process, subsequent block sizes will be restricted to be small, thus leading to complexity savings.

$J_{\hat{\boldsymbol{K}}}(\boldsymbol{w})$ in (2.5) is the maximized projected class mean divided by projected variance in the feature space. $J_{\hat{\boldsymbol{K}}}(\boldsymbol{w})$ increases with the number of features and has no upper limit. Such increase is sometimes due to increasing number of features and does not necessarily improve performance. Thus using $J_{\hat{\boldsymbol{K}}}(\boldsymbol{w})$ by itself for model selection could lead to undesirable results, with a large number of features being chosen. In order to compare all models with different number of features, [38] uses cross validation, an unbiased method, which does not make any assumptions on the data. Despite its advantages, it is time consuming, making it impractical to select a model in large data cases. We propose to use the estimated error rate in (2.8) for each covariance matrix $\hat{\mathbf{K}}$ as a way to compare different candidate structures. The covariance structure with smallest $\hat{P}_e$ will be selected at each step in the sequential search shown in Figure 2.1. Unlike $J_{\hat{\boldsymbol{K}}}(\boldsymbol{w})$ in (2.2), $\hat{P}_e$ is in the range of 0 to 1 for all models. If the data has a Gaussian distribution and means and covariance matrix can be estimated, $\hat{P}_e$ is a good measure of each model's performance. In Section 2.3, experiments on simulated data are used to demonstrate this advantage. Moreover, in experiments on real gene expression data, which is not strictly Gaussian distributed, the measure still generates better results than other algorithms.

2.2.3 Algorithm Improvements

Repeated feature subset selection (RFSS) is applied to reduce the impact of greedy search. RFSS repeats the model construction and feature selection N times.

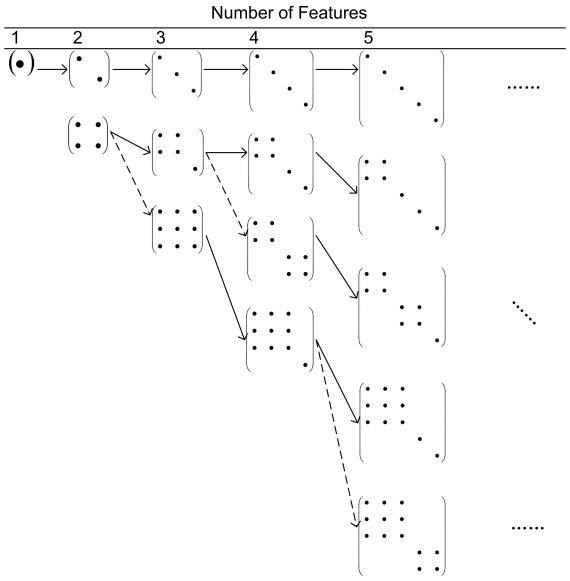


Figure 2.1: Sequential generation of candidate covariance matrix models for BDLDA. Starting with an empty list, we add one feature at a time (namely, the one that maximizes J (2.5)). A feature can be added as an independent block (solid line), or combined with an existing block (dotted line). The best of all these models is selected using $\hat{P}_e$.

At the start, a model with a predefined maximum number of features ($MaxFeature$ in Algorithm 1) is selected. Then the model construction and feature selection is performed again, with the features selected during the first iteration removed from the set of candidate features. This procedure is repeated N times, and each time a feature selection iteration does not consider features already selected in previous iterations. Then the N models are combined by vector concatenating N means and block diagonally concatenating N covariance matrices as shown in Figure 2.2. The feature sets in all N models are different and uncorrelated. The model construction is performed N times or stops when there are not enough candidate features. This improvement enables the algorithm to find more discriminating features without being influenced by previously selected models. The complete algorithm is described in Algorithm 1.

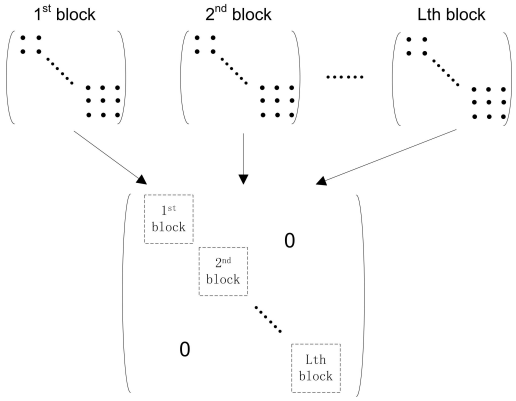


Figure 2.2: Block concatenation procedure

Algorithm 1 Model construction and feature selection

$\mathcal{S} = \emptyset$, $\mathcal{T} =$ all candidate features, $\mathcal{M} = \emptyset$, $F = 0$, $L = 0$

1. Construct the first model (Model 1) by adding feature i, $i = \arg\max_{i \in T} \frac{d_i}{\sigma_i}$ $\mathcal{S} = \mathcal{S} + \{i\}$, $\mathcal{T} = \mathcal{T} - \{i\}$, $\mathcal{M} = \mathcal{M} + \{\text{Model 1}\}$, $F = 1$, $L = 1$
2. For models with feature size F, to create Model j+1, do either of the following
   (1) Add a feature as an independent subblock. The new feature is selected by $i = \arg\max_{i \in T} J$ given in Eq. (2.2) $\mathcal{S} = \mathcal{S} + \{i\}$, $\mathcal{T} = \mathcal{T} - \{i\}$, $\mathcal{M} = \mathcal{M} + \{\text{Model j}\}$, $F = F + 1$, $L = 1$.
   (2) Add a feature to the last subblock if $F < MaxGrow$ and $F + 1$ does not exceed any previous subblocks. The new feature is selected by $i = \arg\max_{i \in T} J$ given in Eq. (2.2). $\mathcal{S} = \mathcal{S} + \{i\}$, $\mathcal{T} = \mathcal{T} - \{i\}$, $\mathcal{M} = \mathcal{M} + \{\text{Model j}\}$, $F = F + 1$, $L = L + 1$.
3. Repeat Step 2 until the $MaxFeature$ is reached.
4. Select among $\mathcal{M}$ the model with minimum P_e given in Eq. (2.8)
5. Remove $\mathcal{S}$ and repeat steps 1-4 N times
6. Combine N selected models

$\mathcal{S}$ is the set of selected features. $\mathcal{T}$ is the set of candidate features. $\mathcal{M}$ is the set of candidate models. F is the number of features in the the model. L is the number of features in the last subblock. $MaxGrow$ is the largest size of a subblock. $MaxFeature$ is the largest number of features in the models.

Prescreening is based on the observation that features with the same means and variances are not discriminating in BDLDA. Some of them may correspond to noise and interfere with classification. Removing these features can improve performance and reduce computation time. A prescreening of all the features is

applied before the model construction in Algorithm 1. The separation of two classes on feature i is represented by $|\frac{d_i}{\sigma_i}|$, with $d_i = m_{Ai} - m_{Bi}$ and $\sigma_i^2 = \frac{1}{K_s}(\sum_{k \in ClassA}(x_{ki} - m_{Ai})^2 + \sum_{k \in ClassB}(x_{ki} - m_{Bi})^2) + c$, where x_{ki} is the ith feature of sample k, K_s is the total number of samples, m_{Ai} is the mean of feature i that belongs to class A and similarly for m_{Bi}, and c is a regularization value.

Only features with $|\frac{d_i}{\sigma_i}|$ above a threshold will be used in Algorithm 1. In the experiments, we use $\frac{1}{3}max_i(|\frac{d_i}{\sigma_i}|)$ as the threshold. To avoid the impact of outliers in real data, instead of using the top ranking $|\frac{d_i}{\sigma_i}|$, the average of 10 largest $|\frac{d_i}{\sigma_i}|$ is used, that is, the threshold is one third of the average of 10 largest $|\frac{d_i}{\sigma_i}|$. The prescreening procedure can also be applied to other classification tools.

2.2.4 Treelets

Though BDLDA has high classification accuracy as compared to some state-of-art algorithms, it can neglect useful correlations in the covariance matrix, because we limit the block size to be not larger than 3. However increasing the block size increases the complexity of BDLDA. As mentioned in the introduction, if the covariance matrix is transformed to be diagonal by PCA, an ideal low dimensional approximation in terms of variance can be achieved. However, this cannot be done because of the large feature size.

For gene expression data analysis, many methods have been proposed to identify groups of highly correlated features and select a few representative features for each group. Treelets [31] can be used to identify subsets of genes that exhibit correlation in their expression patterns, but will replace each such localized group by a linear combination that encodes the information from all features in that group.

At each level of the tree, the two most correlated features, say feature α and feature β, are transformed and replaced by two transformed coefficients, representing their projection into their principal component and the vector orthogonal to the principal component respectively. The correlation between two features is defined as the Pearson's correlation, which is derived by dividing the covariance of the two features by the product of their standard deviations. Treelets is an unsupervised feature transformation measure that does not consider class information but can successively exploit correlation between features The treelets algorithm is described in Algorithm 2.

Algorithm 2 Treelets Algorithm

1. Compute the sample covariance and correlation matrix $\hat{\boldsymbol{\Sigma}}^{(0)}$ and $\hat{\boldsymbol{M}}^{(0)}$ from original data. Initialize the set of indices $\delta = \{1, 2, \cdots, p\}$. Each index represents a tree branch that can be combined into a higher level. Initially, δ contains every original feature. Define $\boldsymbol{B}^{(0)}$ as an identity matrix.
Repeat 2-5 for $l = 1, \cdots, L$,
2. Find the two most similar features from δ according to the correlation matrix $\hat{\boldsymbol{M}}^{(l-1)}$. Let $(\alpha, \beta) = \arg\max_{i,j \in \delta} \hat{\boldsymbol{M}}^{(l-1)}$, where $i < j$.
3. Perform a local PCA on (α, β). Find a Jacobi rotation matrix $\boldsymbol{J}(\alpha, \beta, \theta_l)$, $|\theta_l| \leq \pi/4$ and $\hat{\boldsymbol{\Sigma}}^{(l)}_{\alpha\beta} = \hat{\boldsymbol{\Sigma}}^{(l)}_{\beta\alpha} = 0$.
4. Update the following matrices $\hat{\boldsymbol{\Sigma}}^{(l)} = \boldsymbol{J}^T \hat{\boldsymbol{\Sigma}}^{(l-1)} \boldsymbol{J}$, $\boldsymbol{B}^{(l)} = \boldsymbol{B}^{(l-1)} \boldsymbol{J}$, $\boldsymbol{x}^{(l)} = \boldsymbol{J}^T \boldsymbol{x}^{(l-1)}$. Update $\hat{\boldsymbol{M}}^{(l)}$ accordingly.
5. Retain the principal component α in δ to represent the branch and remove β, $\delta = \delta \backslash \{\beta\}$.

At level l, an original sample $\boldsymbol{x}$ can be written as

$$\boldsymbol{x} = \boldsymbol{B}^{(l)} \boldsymbol{x}^{(l)} \tag{2.9}$$

where each row vector of $\boldsymbol{B}^{(l)}$ is a basis at treelets level l. $\boldsymbol{x}^{(l)}$ is the projection of $\boldsymbol{x}$ onto those basis vectors. The objective is to reduce the amount of off diagonal energy in the covariance matrix obtained from $\boldsymbol{x}^{(l)}$, i.e., in the transformed space.

Each feature in $\boldsymbol{x}^{(l)}$ is a linear combination of several features in $\boldsymbol{x}$. At a fixed level $\mathcal{L}$, $\boldsymbol{x}^{(\mathcal{L})}$ is the sample to be be used as an input to BDLDA. Note that in general $\boldsymbol{x}^{(\mathcal{L})}$ will contain features that have not been modified, i.e., they are the same as in $\boldsymbol{x}$, while some features may be the result of applying a transformation to multiple inputs in the original feature vector. For example, one feature in $\boldsymbol{x}^{(\mathcal{L})}$ may contain information of at most $2^{\mathcal{L}}$ features in $\boldsymbol{x}$ when an $\mathcal{L}$-level treelet is used. In this case $\boldsymbol{B}^{(l)}$ would include a column with $2^{\mathcal{L}}$ non-zero values.

2.2.5 Algorithm Description

T-BDLDA combines treelets and BDLDA. First, we build a treelets basis matrix $\boldsymbol{B}^{(\mathcal{L})}$ at a certain level $\mathcal{L}$ using training data. Each training sample $\boldsymbol{x}$ is transformed into $\boldsymbol{x}^{(\mathcal{L})}$ as in (2.9). The $\boldsymbol{x}^{(\mathcal{L})}$ are taken as inputs into BDLDA to build a classification model. Each test sample $\boldsymbol{y}$ is also projected as in (2.9) onto the basis $\boldsymbol{B}^{(\mathcal{L})}$ obtained from training samples. The treelets coefficients of test samples $\boldsymbol{y}^{(\mathcal{L})}$ were used in testing in BDLDA.

Note that even if we use the same number of blocks of the same size as when applying BDLDA to original expression data, we would be essentially selecting more actual features and thus larger effective blocks. The resulting improvements in classification results could alternatively be attained by using more blocks with potentially larger size in BDLDA without using the treelets transformation. However, increasing the number and size of the blocks can be computationally expensive in BDLDA. Thus T-BDLDA provides a method to effectively find more discriminating features and their structure in affordable time.

The complete T-BDLDA algorithm is summarized in Algorithm 3

Algorithm 3 Complete Algorithm of treelets and BDLDA

1. Threshold data by t-score.
2. Obtain treelets basis on training data at level $\mathcal{L}$.
3. Transform both training and test data by treelets basis by (2.9).

Run BDLDA on treelets coefficients of training data .

4. Build candidate blocks of size $\mathcal{M}axFeature$ by adding features according to (2.2). Each subblock inside candidate blocks has maximum size $\mathcal{M}axGrow$.
5. Select the block using (2.8).
6. Remove previously selected blocks and repeat steps 4 and step 5 $\mathcal{N}$ times.
7. Combine $\mathcal{N}$ selected blocks.
8. Test the treelets coefficients of test samples by the selected BDLDA model.

$\mathcal{L}$ the level of treelets to extract coefficients. $\mathcal{M}axGrow$ is the largest size of a subblock. $\mathcal{M}axFeature$ is the largest number of features in a block. $\mathcal{N}$ is the number of blocks.

2.3 Experimental Results

Both BDLDA and T-BDLDA are tested on both simulated data and real data. The results are compared with SeqDLDA [39], NSC [52] and SCRDA [19]. The test results (both error rates and standard deviation) shown in Table 2.1 and Table 2.2 are obtained by performing 10 fold cross validation 50 times. The error rates of SCRDA in [19] are presented as a matrix according to two tuning parameters. The smallest error rate among all parameter pairs is shown.

2.3.1 Simulated Data

Block diagonal covariance matrix The distributions of two classes are $N(\boldsymbol{\mu_A}, \boldsymbol{K})$ and $N(\boldsymbol{\mu_B}, \boldsymbol{K})$ with total number of features $P = 10000$, and with $\boldsymbol{\mu_A} = (\underbrace{00 0 \cdots 00}_{10000})$, and $\boldsymbol{\mu_B} = (\underbrace{0.5 \cdots 0.5}_{200} \underbrace{00 \cdots 00}_{9800})$. The block diagonal structure of $\boldsymbol{K}$ is shown in (2.10). Each subblock has an autoregressive structure, which is a symmetric Toeplitz matrix with the first row set to $(1 \ \rho \ \cdots \ \rho^{98} \ \rho^{99})$. The subblock size is 100×100 and there are a total of 100 subblocks. It is assumed the autocorrelation

within each subblock is $|\rho| = 0.9$ and we set alternating signs for each subblock. 220 samples are generated. The average error rates and standard deviations are shown in Table 2.1.

$$\mathbf{K} = \begin{pmatrix} \boldsymbol{K_\rho} & 0 & 0 & \ddots & \ddots & \ddots \\ 0 & \boldsymbol{K_{-\rho}} & 0 & 0 & \ddots & \ddots \\ 0 & 0 & \boldsymbol{K_\rho} & 0 & \ddots & \ddots \\ \ddots & 0 & 0 & \boldsymbol{K_{-\rho}} & 0 & \ddots \\ \ddots & \ddots & \ddots & 0 & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}_{10000\times 10000} \tag{2.10}$$

Diagonal covariance matrix The distributions of two classes are $N(\boldsymbol{\mu_A}, \boldsymbol{K})$ and $N(\boldsymbol{\mu_B}, \boldsymbol{K})$ with total number of features $P = 10000$, $\boldsymbol{\mu_A} = (\underbrace{000\cdots00}_{10000})$, and $\boldsymbol{\mu_B} = (\underbrace{0.5\cdots0.5}_{100}\underbrace{00\cdots00}_{9900})$. We assume that features are independent so that the covariance can be written, $\boldsymbol{K} = \boldsymbol{I_P}$, where $\boldsymbol{I_P}$ is the $P \times P$ identity matrix. 220 samples are generated.

Toeplitz covariance matrix The distributions of two classes are $N(\boldsymbol{\mu_A}, \boldsymbol{K})$ and $N(\boldsymbol{\mu_B}, \boldsymbol{K})$ with total number of features $P = 1000$. The difference of means are assumed to be fading exponentially. $\boldsymbol{\mu_A} = (\underbrace{000\cdots00}_{1000})$. $\boldsymbol{\mu_{Bj}} = e^{-\gamma j}, (j = 1, 2\cdots1000)$. $\gamma = 0.05$. It is assumed that $\boldsymbol{K}$ is the Toeplitz matrix with the first row $(1\ \frac{-1}{2}\ \frac{2}{5}\ 0\ \cdots\ 0)$. 120 samples are generated.

2.3.2 Real Data

We test our algorithm on two-class cancer data, using two publicly available online for colon cancer (62 samples, 2000 features) [15] and prostate cancer (102 samples, 6033 features) [14]. 10 fold cross validation is done 50 times on each data set.

Table 2.1: Average Error Rate (Standard Deviation) for Simulated Data

	Block Diagonal covariance	Diagonal covariance	Toeplitz covariance
T-BDLDA	**<0.45%** (0%)	3.41% (1.14%)	**3.17%** (0.95%)
BDLDA	0.36% (0.19%)	3.57% (1.09%)	4.51% (1.02%)
SeqDLDA	19.64% (1.5%)	2.57% 0.83%	8.97% (1.71%)
NSC	18.15% (1.34%)	6.89% (1.12%)	10.82% (1.74%)
SCRDA	9.45% (1.23%)	**1.97%** (0.62%)	10.2% (1.37%)
$\mathcal{L}$=5, $\mathcal{M}axGrow$=3, $\mathcal{M}axFeature$=20, $\mathcal{N}$=5			

Table 2.2: Average Error Rate (Standard Deviation) for Real Data

	Colon Cancer	Prostate Cancer
T-BDLDA	**9.84%** (0.51%)	**5.1%** (0.62%)
BDLDA	10.06% (1.15%)	5.21% (0.85%)
SeqDLDA	12.06% (1.87%)	5.53% (0.9%)
NSC	10.31% (1.02%)	7.65% (0.42%)
SCRDA	11.41% (1.69%)	5.41% (0.89%)
$\mathcal{L}$=5, $\mathcal{M}axGrow$=3, $\mathcal{M}axFeature$=20, $\mathcal{N}$=5		

2.3.3 Discussion

In simulated data with block diagonal covariance matrix and Toeplitz covariance matrix, T-BDLDA performs the best, and BDLDA is the second best. In block diagonal covariance matrix, the simulation error is 0 ($<1/220$ in Table 2.2). For diagonal covariance matrix, our algorithms do not show much advantage over SeqDLDA and SCRDA, which are DLDA based methods. T-BDLDA outperforms BDLDA in all simulated data. This demonstrates the advantage of using treelets coefficients instead of raw data. The margin we observe in data with diagonal covariance matrix may come from selecting more actual features involved in the treelets coefficients in the selected blocks. The gain in the other two data sets with correlations may come from the energy concentrating on diagonal terms, which is beneficial for block diagonal structure in classification. In the two real data sets, T-BDLDA has the lowest error rates. The promising results of our work on T-BDLDA shows that it can be an competitive algorithm for classification of gene expression data.

2.4 Conclusions

We have proposed and improved a supervised learning algorithm, BDLDA, for high dimensional data, typically for RNA gene expression data. Though BDLDA has superior classification accuracy compared with state-of-art algorithms, it still misses a lot of information in the off-diagonal positions of the covariance matrix. To better improve the classification performance, treelets is proposed as a preprocessing step to be applied to gene expression data leading to a covariance matrix having less off diagonal energy in covariance matrix. Then the treelets coefficients are input into BDLDA. A block diagonal structure is imposed on the covariance matrix

with predefined number of blocks and block size. RFSS and prescreening are used to improve the algorithm. T-BDLDA outperforms BDLDA, SeqDLDA, NSC and SCRDA in most simulated and both real data used in our tests. T-BDLDA is promising to handle data with small number of training samples, a very large number of features and an unknown correlation structure.

Chapter 3

Graph Based One Class Learning

3.1 Introduction

In recent years, the graph Laplacian has become an important tool in manifold related machine learning algorithms [5] [58] [63] [4] [62] [8]. In these algorithms, data points are represented by graph nodes and similarity between data points by edge weights. The goal is to infer labels on nodes for which no label data is known. The algorithms can be viewed as estimating a classification function **F** on the graph, where **F** needs to satisfy two criteria simultaneously: (1) it should take a value close to that corresponding to the initial labels, and (2) it should be smooth, i.e., neighboring nodes in the graph should have similar values. This can be expressed in a regularization framework where the first term is a loss function of the labeled data, and the second term is a regularizer of the graph structure and the label function. The various manifold learning algorithms differ in the particular choice of the loss function and the regularizer, but have in common some parameters that control the balance between the two terms. However, only a limited amount of work has been devoted to discussion of how these regularization parameters should be chosen.

Among the manifold learning algorithms, learning with local and global consistency (LGC) [58] has a closed form solution. It has been widely used for semi-supervised learning (SSL). A key aspect of LGC is to let every data point iteratively spread its label information to its neighbors until a global stable state is achieved. LGC can be explained from the perspective of lazy label propagation. Each row in $\mathbf{F}$ is the probability of a data point belonging to each class. The probability is propagated through the graph according to edge weights, while there remains a certain probability of keeping the initial labels. The balance of label propagation and initial label fit is controlled by a parameter, which we denote here as α. This will be formally introduced in Section 3.2.

One application of LGC is classification, which is performed by examining each row of the classification function at convergence. This framework leads to two questions: 1. Is the algorithm sensitive to the choice of α? 2. How does the ratio of labeled data in different classes affect the classification accuracy? We show, in both simulated and real data, that classification results, to a large extent, depend on the choice of α, especially when the labeled data are unbalanced. Thus, in LGC classification, an optimal α has to be selected by validation. Validation requires setting aside a separate validation data set, which is not a problem when large amount of data is available. When the amount of data is limited, however, validation has to be done using time consuming methods, such as cross validation or boostrapping.

Another major application of LGC is the universal ranking problem [60]. The LGC ranking problem has the same graph structure and label spreading process as LGC classification. When the spreading process is repeated until convergence, the samples are ranked according to a particular column j of $\mathbf{F}$ at convergence, which corresponds to samples' probability of belonging to class j. In other words, LGC

ranks the samples by their class preference. In this thesis, we prove that it is possible to select the parameter for LGC ranking without validation. Since validation either requires extra number of samples, or much more time to run validation methods, such as cross validation, it is desirable to select a parameter value that theoretically guarantees a stable and optimal or suboptimal ranking. For a particular ranking problem, there is a bound on the parameter α. If α is below the bound, the ranking is theoretically stable and experimentally close to optimal. Thus we can choose any α below the bound. Time consuming validation methods, such as cross validation or bootstrapping are unnecessary in LGC ranking.

In this work, we define the pairwise ranking of two samples to be stable if the ranking remains the same when α changes. We prove that the ranking is stable if α is below a bound. The pairwise ranking of two samples, which is also the difference between two rows of the classification function $\mathbf{F}$ at convergence, can be expressed in terms of α and the entries of the similarity matrix. It is hard to decompose α and the entries of the similarity matrix, so that the influence of changing α on the ranking is unclear. When α is small enough, the pairwise ranking can be approximated by a product of a function of α and a function of the entries of the similarity matrix. By doing this, we have separated α and the similarity matrix, which is independent of α. When α changes only below this bound, the pairwise rankings do not change. In order for the approximations to hold, α needs to be below a bound, which depends on the similarity matrix. In simulations of LGC ranking on various data sets, including the two-moon simulated data and eight real data sets, the ranking accuracy of smaller α is consistently higher than that of larger α, and the standard deviation of accuracy is consistently lower. This demonstrates that LGC ranking becomes more stable and accurate when α is below the bound.

As an application of LGC ranking, we propose a one class learning algorithm, which we call, Graph-OCL. One class learning is a type of binary classification in which only one class has labeled data, so most of state-of-art supervised classifiers do not apply. One example of one class learning is a situation where we have training samples of research papers and we want to retrieve similar papers from the web. We solve the one class learning problem in two steps. Step one is identification of reliable negative samples. In this step, we use LGC ranking and select the samples with highest ranking probability of belonging to negative class. Step two is classification with the initially labeled data and the identified negative samples. In this step, we use transductive SVM (TSVM) [23] as a classifier. We test Graph-OCL on Newsgroup 20 data set. The proposed algorithm is demonstrated to be more effective than Naive Bayesian (NB) and S-EM [33], an improved version of NB.

This chapter is organized as follows. Section 3.2 provides the definitions and algorithm description of LGC. Section 3.3 provides experiments on how the ratio of labeled data in different classes and the choice of α influence LGC classification accuracy. Section 3.4 provides theorems proving that ranking is stable if α is below a bound. We also discuss how the bound is calculated. Section 3.5 provides an application of LGC ranking to one class learning. Section 3.6 concludes the chapter.

3.2 Brief Overview of LGC

We first introduce LGC algorithm [58] [60], then explain the algorithm from the perspective of label information propagation.

3.2.1 LGC Algorithm

Consider a sample set $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\} \subset \mathbf{N}^m$, where each sample is a vector of features. Without loss of generality, let the first l samples $\mathbf{x}_i$ $(1 \leq i \leq l)$ be labeled and the remaining ones $\mathbf{x}_u$ $(l+1 \leq u \leq n)$ be unlabeled. We build a graph, in which the nodes are labeled and unlabeled samples, and the affinity matrix $\mathbf{W}$ quantifies the similarity between the nodes, in terms of distance in the feature space.

Let $\mathbf{F}(t)$ be a $n \times c$ matrix, that corresponds to a classification of $\mathcal{X}$ at time $t(t = 0, 1, \cdots, +\infty)$. c is the number of classes. $F_{ij}(t)$ represents the probability that at time t, instance i has label j:

$$F_{ij}(t) = P(\text{instance } i \text{ has label } j,\ t) \qquad j = 1, 2, \cdots, c \tag{3.1}$$

Each sample $\mathbf{x}_i$ can then be assigned a label $y_i = \max\{F_{i1}(t), F_{i2}(t), \cdots, F_{ic}(t)\}$ $(\sum_{k=1}^{c} F_{ik}(t) = 1,\ 0 \leq F_{ik}(t) \leq 1,\ 1 \leq k \leq c)$.

At the start of the iteration, assume $F_{ik}(0) = 1$ if the initial label of $\mathbf{x}_i$ is k and $F_{ik}(0) = 0$ otherwise. Assume $F_{ik}(0) = 1/c$ if $\mathbf{x}_i$ is unlabeled. With this, we can define $\mathbf{Y} = \mathbf{F}(0)$. $\mathbf{F}(t)$ represents the labeling matrix at each step of iteration while $\mathbf{Y}$ represents the initial labels.

Construct matrix $\mathbf{S} = \mathbf{D}^{-1}\mathbf{W}$ in which $\mathbf{D}$ is the diagonal degree matrix with its (i, i)-entry equal to the sum of the i-th row of $\mathbf{W}$. We then iterate

$$\mathbf{F}(t+1) = \alpha\mathbf{S}\mathbf{F}(t) + (1-\alpha)\mathbf{Y} \tag{3.2}$$

until convergence, where α is a predefined parameter with $\alpha \in (0, 1)$. [58] shows that $\mathbf{F}(t)$ converges as follows. Since $\mathbf{F}(0) = \mathbf{Y}$ and given (3.2), we have

$$\mathbf{F}(t) = (\alpha\mathbf{S})^{t-1}\mathbf{Y} + (1-\alpha)\sum_{i=0}^{t-1}(\alpha\mathbf{S})^{i}\mathbf{Y} \tag{3.3}$$

Since $0 < \alpha < 1$ and the eigenvalues of $\mathbf{S}$ are in $[-1, 1]$,

$$\lim_{t\to\infty}(\alpha\mathbf{S})^{t-1} = 0 \text{ and } \lim_{t\to\infty}\sum_{i=0}^{t-1}(\alpha\mathbf{S})^{i} = (I-\alpha\mathbf{S})^{-1}. \tag{3.4}$$

Let $\mathbf{F}^*$ denote the limit of the sequence $\mathbf{F}(t)$ as $t \to \infty$. Then we can derive a closed form expression:

$$\mathbf{F}^* = (1-\alpha)(\mathbf{I}-\alpha\mathbf{S})^{-1}\mathbf{Y} \tag{3.5}$$

If LGC is applied to a classification problem, for sample i, we then select class k which has the maximum likelihood, i.e., $\max_k F^*_{ik}, (1 \le k \le c)$.

The other application of LGC is ranking, in which the order of a series of test samples is learned, and the order is consistent with the known ranking in the training data. Usually the LGC ranking could be with respect to one of the classes, e.g., class k. We rank the kth column of $\mathbf{F}^*$, which is the relevance score corresponding to the query points.

The LGC classification and ranking algorithm are described in Algorithm 4.

Algorithm 4 LGC classification/ranking

1. Form the similarity matrix $\mathbf{W}$.
2. Construct matrix $\mathbf{S} = \mathbf{D}^{-1}\mathbf{W}$ in which $\mathbf{D}$ is a diagonal matrix with its (i, i)-entry equal to the sum of the i-th row of $\mathbf{W}$.
3. Calculate $\mathbf{F}^* = (1-\alpha)(\mathbf{I}-\alpha\mathbf{S})^{-1}\mathbf{Y}$.
4. (classification). Classify sample as class k which has $\max_k F^*_{ik}$.
4. (ranking). Rank the kth column of $\mathbf{F}^*$.

3.2.2 Interpretation of LGC

We interpret Algorithm 4 in terms of a lazy information transfer network. The idea is similar to the lazy random walk introduced in [59], but has a probability interpretation of the label functions.

Section 3.2.1 defines the initial state $\mathbf{F}(\mathbf{0})$ as $F_{ik}(0) = 1$ if the initial label of $\mathbf{x}_i$ is k and $F_{ik}(0) = 0$ otherwise, while if $\mathbf{x}_i$ is unlabeled, we assume $F_{ik}(0) = 1/c$. $F_{ik}(0) = 1$ means sample $\mathbf{x}_i$ has probability 1 to belong to class k. For those unlabeled, $F_{i1}(0) = \cdots = F_{ic}(0) = 1/c$ represents our ignorance of its probability at the start of the algorithm. The probability of belonging to c classes propagates over time based on the lazy information transfer network. From (3.2),

$$F_{ij}(t+1) = \alpha \sum_{z=1}^{n} S_{iz} F_{zj}(t) + (1-\alpha) Y_{ij} \tag{3.6}$$

$\mathbf{S}$ is a row stochastic matrix, which can be seen as the information transfer probability matrix. We interpret S_{iz} as the proportion of information that $F_{ij}(t+1)$ obtains from $F_{zj}(t)$ ($1 \leq z \leq n$ and $j = 1, 2, \cdots, c$). It is similar to the random walk, because both interpretations of S_{iz} represent the similarity between $\mathbf{x}_i$ and $\mathbf{x}_z$. However, in the random walk S_{iz} is the probability of state change from $\mathbf{x}_i$ to $\mathbf{x}_z$, while in information transfer network, S_{iz} is the probability of information transfer from $\mathbf{x}_z$ to $\mathbf{x}_i$.

This information transfer network is called lazy, because $\mathbf{F}(t)$ also depends on the initial state $\mathbf{Y}$. This means that at each iteration, each sample in the graph has probability $1-\alpha$ of retaining its initial label. With the probability α, a sample changes label with the probability proportional to the weight of the link. We assume that information transfer happens within each label and not across labels, which

means that $F_{ij}(t)$ depends on $F_{zj}(t-1)$, where $1 \leq z \leq n$, and $F_{ij}(t)$ does not depend on $F_{zv}(t-1)$, where $v \neq j$.

F^*_{ij} $(j = 1, 2, \cdots, c)$ is the probability of sample $\mathbf{x}_i$ having label j when the algorithm converges. In LGC classification, we compare the probability of belonging to each class, and choose the class that has the highest probability. In LGC ranking, we sort the probability of samples according to the kth column of $\mathbf{F}$, i.e., $F^*_{\cdot k}$, which represents each data's relevance to the labeled data.

Sometimes, $\mathbf{S}$ is normalized as a symmetric matrix as $\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. The LGC classification function $\mathbf{F}$ can be rewritten using the symmetric graph Laplacian $\mathbf{L_s} = \mathbf{I} - \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. $\mathbf{L_s}$ was used for the purpose of classification in [58]. However, $\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ is not a stochastic matrix, so the sum of each row of $\mathbf{F}^*$ is no longer 1. Thus $\mathbf{F}$ does not have a probability interpretation corresponding to each of its rows. Though we can still perform classification according to the sample's class preference in the corresponding row of $\mathbf{F}$, ranking is difficult since we do not have a general interpretation corresponding to each column of $\mathbf{F}$. Thus, in the following analysis and applications, $\mathbf{S}$ is defined as $\mathbf{D}^{-1}\mathbf{W}$.

3.2.3 Choice of α

The classification function at convergence $\mathbf{F}^*$ is the solution to the following optimization function:

$$\mathbf{F}^* = \arg\min_{\mathbf{F}} \frac{1}{2}(\alpha \sum_{i,j=1}^{n} S_{ij} \parallel F_i - F_j \parallel^2 -(1-\alpha)\sum_{i=1}^{n} \parallel F_i - Y_i \parallel^2) \tag{3.7}$$

Let $Q(\mathbf{F}) = \frac{1}{2}(\alpha \sum_{i,j=1}^{n} S_{ij} \parallel F_i - F_j \parallel^2 -(1-\alpha)\sum_{i=1}^{n} \parallel F_i - Y_i \parallel^2)$, which is the function we want to optimize. $Q(\mathbf{F})$ can be written in the matrix form:

$$Q(\mathbf{F}) = \alpha(\mathbf{F}^T\mathbf{F} - \mathbf{F}^T\mathbf{S}\mathbf{F}) + (1-\alpha)(\mathbf{F}-\mathbf{Y})^T(\mathbf{F}-\mathbf{Y}) \tag{3.8}$$

Taking the derivative of $Q(\mathbf{F})$ with respect to $\mathbf{F}$ and making it equal to 0, we can derive the solution as $\mathbf{F}^* = (1-\alpha)(\mathbf{I}-\alpha\mathbf{S})^{-1}\mathbf{Y}$.

α is the only parameter used in LGC. [61] discusses α's impact on the ranking when $\alpha \approx 1$. However, α's impact on classification or ranking when $\alpha \in (0.1)$ has not been fully studied so far. In (3.7), α controls the balance between the graph energy (similar graph nodes should have similar labels) and initial label fit ($\mathbf{F}^*$ should be close to $\mathbf{Y}$). Consider two extreme cases, i.e., $\alpha = 1$ and $\alpha = 0$. $\alpha = 1$ leads to $\mathbf{F}(t) = \mathbf{S}^t\mathbf{F}(0)$. The quantity $\lim_{t\to\infty} S^t(i,j)$ is proportional to the degree of the node, which means every row of $\lim_{t\to\infty}\mathbf{S}^t$ is the stationary distribution of the information transfer network. Thus, through iteration, the initial labels cannot propagate and are not considered as generating the final class probabilities.

The other extreme case is that of $\alpha = 0$, which implies that the probability distribution of labels does not change along time. Therefore, the first term $\alpha\mathbf{S}\mathbf{F}(t)$ in the iteration function (3.2) represents propagating information along the data structure, and the second term $(1-\alpha)\mathbf{Y}$ imposes a regularization term, so that the classification conforms to the initial labels. However, it is not obvious how to choose α in the range $0 < \alpha < 1$. In the following sections, we analyze α's influence on LGC classification and ranking.

3.3 LGC Classification

As mentioned in the introduction, when applied to classification, there are two unsolved questions with respect to LGC: (1) Given the similarity matrix, is the algorithm sensitive to the choice of α? (2) How does the ratio of labeled data among

different classes affect the classification accuracy? We test the LGC classification on two real data sets, the USPS digit recognition data and the Mnist data, in order to empirically answer the two questions. Each data set contains 10 subsets, which correspond to 10 digits. For each data set, we use one subset as positive data and one other subset as negative data, and let α vary between 0 and 1 $(0.1, 0.2, \cdots, 0.9)$. We use the RBF kernel as similarity matrix in both data sets.

Figures 3.1 and 3.2 show the classification accuracy of the subsets of the USPS and the Mnist data, from which we have two observations.

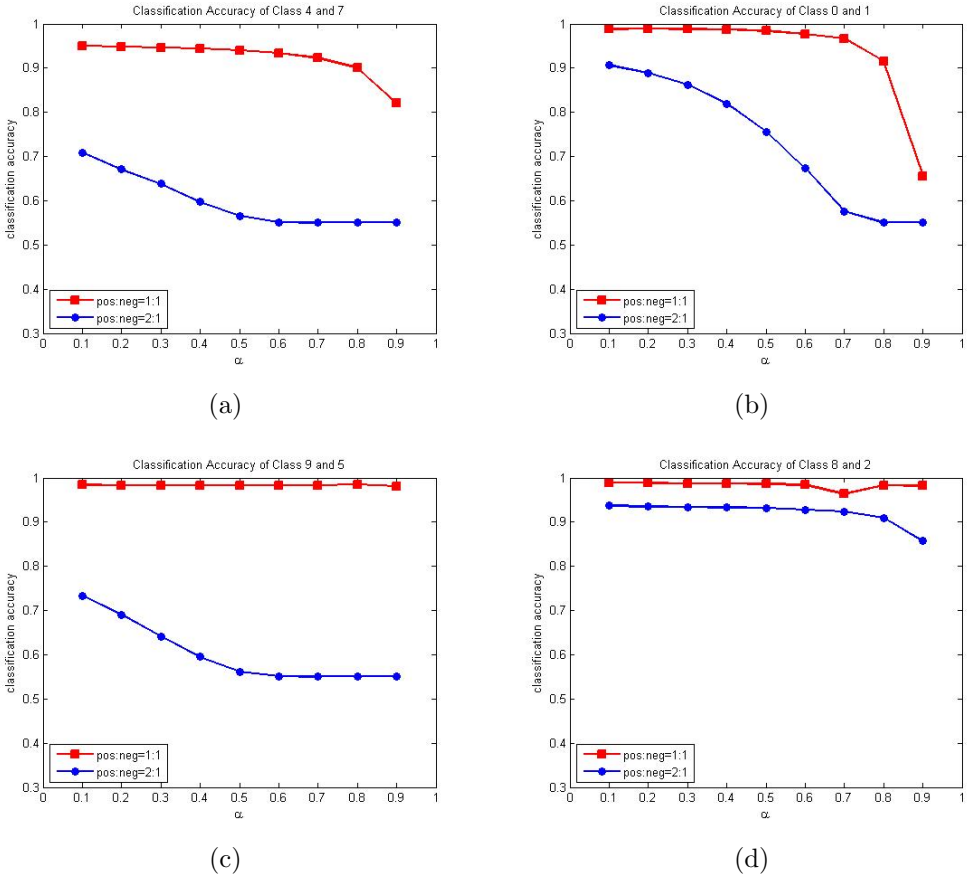


Figure 3.1: Classification accuracy of subsets of USPS data. The ratio of labeled data in positive class to negative class in each subgraph is 1:1 and 2:1. (a) USPS Digit 4 and 7; (b) USPS Digit 0 and 1; (c) USPS Digit 9 and 5; (d) USPS Digit 8 and 2;

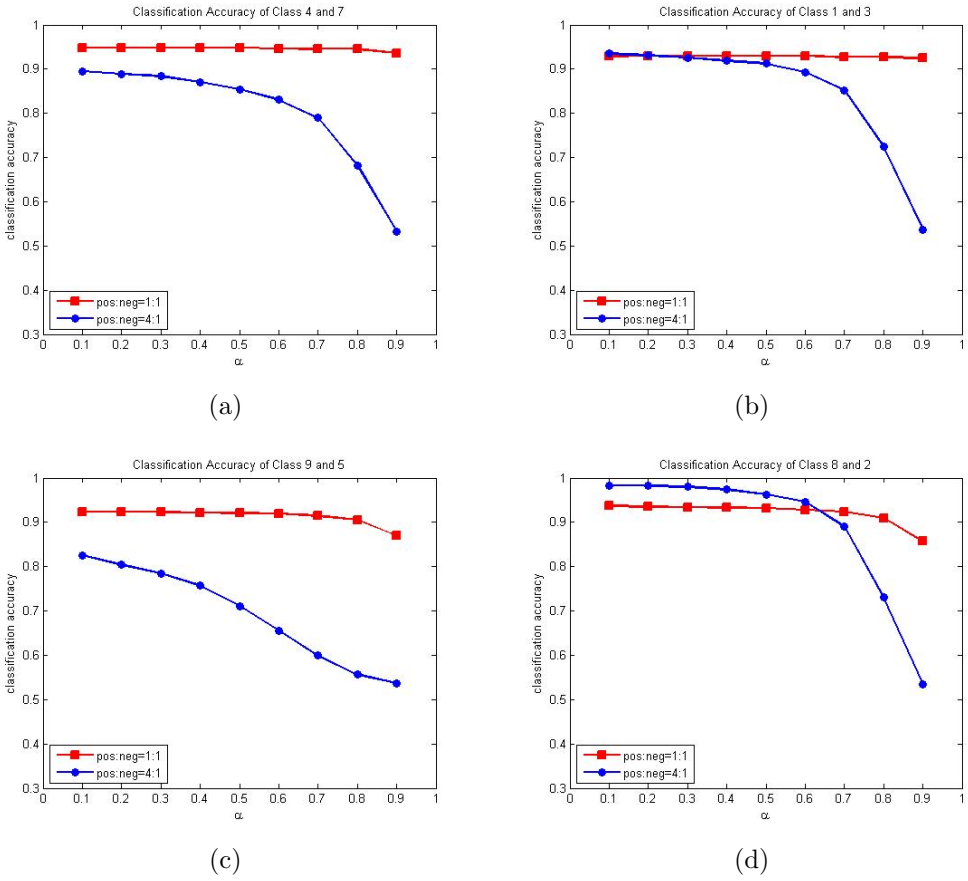


Figure 3.2: The classification accuracy of subsets of Mnist data. The ratio of labeled data in positive class to negative class in each subgraph is 1:1 and 4:1. (a) Mnist Digit 4 and 7; (b) Mnist Digit 1 and 3; (c) Mnist Digit 9 and 5; (d) Mnist Digit 8 and 2;

Observation 1: The classification accuracy is sensitive to the ratio of labeled data among different classes. The classification is obtained by propagating the label information through time according to (3.6). In case of unbalanced labeled data, which means the ratio of labeled data is far from 1 among different classes, the information of the class with majority of labeled data propagates more label information than the other class. Thus, the unlabeled data has a preference for the class with more labeled data.

Observation 2: α has an impact on the classification accuracy, and the impact is greater when the labeled data is unbalanced. α controls the trade-off between the graph energy and initial label consistency. In case of balanced labeled data, the initial labels are consistent with graph structure, thus changing α does not change the accuracy very much. However, when the labeled data is unbalanced, giving more weight to the graph energy helps propagating label information according to (3.6). More labeled information of the class with a higher number of labeled data is propagated. This will reduce classification accuracy if a constant threshold is used.

The above observations show that LGC is sensitive to α, especially when labeled data is unbalanced among classes. Thus, when LGC is applied to classification, especially the labeled data in different classes are unbalanced, α must be carefully chosen by using validation methods.

3.4 Analysis of LGC for the Ranking Problem

Usually unbalanced labeled data among different classes is not a problem in LGC ranking, because the orders of probability functions of data points, rather than their absolute values, are important, and there is no threshold to classify the data. [60] and [45] have shown the promise of using LGC as a ranking algorithm if only one class has initially labeled data. [45] also shows experimentally that the ranking is insensitive to α when α is between 0.1 and 0.9, when used as first step in one class learning.

In this section, we first present previous work, which shows that ranking based on Laplacian matrix is sensitive to α, when α is close to 1. Then we present our analysis of the general case $0 < \alpha < 1$ in LGC ranking.

3.4.1 Analysis for the case $\alpha \approx 1$

Zhou et al. [61] show that the LGC ranking function (when defining $\mathbf{S}$ as $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$) is sensitive to α if $\alpha \approx 1$ by using Green's function of a Laplace operator. They also show that the ranking function is only influenced by the way the graph is constructed in case of α close to 1. We prove that LGC ranking has a similar property if $\alpha \approx 1$ when $\mathbf{S}$ is defined as a stochastic matrix $\mathbf{S} = \mathbf{D}^{-1}\mathbf{W}$.

When used in ranking, the constant $(1-\alpha)$ term in (3.5) can be dropped, then the ranking function becomes $\mathbf{F}^* = (\mathbf{I}-\alpha\mathbf{S})^{-1}\mathbf{Y}$. Let $\mathbf{L}_s = \mathbf{I}-\mathbf{S}$ and $\beta = \frac{1-\alpha}{\alpha}$, then $\mathbf{F}^* = \frac{1}{\alpha}(\beta\mathbf{I} + \mathbf{L}_s)^{-1}\mathbf{Y}$. Denote $\mathbf{G}_s = (\beta\mathbf{I} + \mathbf{L}_s)^{-1}$. By eigenvector decomposition,

$$\mathbf{G}_s = \sum_{k=1}^{n} \frac{1}{\lambda_k + \beta}\mathbf{v}_k\mathbf{v}_k^{-1} \tag{3.9}$$

where λ_k is the kth eigenvalue of $\mathbf{L}_s$, and $|\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_n|$. By Perron-Frobenius Theorem, $|\lambda_k| \leq 2, (1 \leq k \leq n)$. The eigendecomposition of $\mathbf{L}_s$ can be written as $\mathbf{L}_s = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$, where $\mathbf{v}_k$ is the eigenvector associated with λ_k, and the kth columns of $\mathbf{V}$. $\mathbf{v}_k^{-1}$ is the kth row of $\mathbf{V}^{-1}$. Since $\lambda_1 = 0$, $\mathbf{G}_s$ can be written as:

$$\mathbf{G}_s = \frac{1}{\beta}\mathbf{v}_1\mathbf{v}_1^{-1} + \sum_{k=2}^{n} \frac{1}{\lambda_k + \beta}\mathbf{v}_k\mathbf{v}_k^{-1} \tag{3.10}$$

If β is small enough such that $|\frac{1}{\beta}| \gg |\frac{1}{\lambda_i+\beta}|, (2 \leq i \leq n)$, the behavior of the ranking function is determined by $\mathbf{v}_1$ alone, where

$$\mathbf{F}^* \approx \frac{1}{\beta}\mathbf{v}_1\mathbf{v}_1^{-1} \tag{3.11}$$

and

$$\mathbf{v_1} = (1, 1, \cdots, 1)^T \quad \text{and} \quad \mathbf{v_1^{-1}} = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})^T \tag{3.12}$$

This implies that for small β, the ranking function is only influenced by the size of the graph, and does not depend on the initial label. This is consistent with our analysis of the effect of α on the tradeoff of graph energy and initial label fit. It is also worth noticing that each row in $\mathbf{F}^*$ is approximately the same if $\alpha \approx 1$, which means that there is no effective ranking.

It is still unclear how the ranking function is generally impacted by α, if α is not close to 1. In the following, we discuss the general case of $\alpha, (0 < \alpha < 1)$ and its impact on the ranking function.

3.4.2 Analysis of LGC Ranking

3.4.2.1 Ranking Decomposition

To simplify the analysis, we assume the number of classes is $c = 2$. The analysis can be extended to multiple classes. We also assume the queries (labeled data) are in class 1, without loss of generality. We want to prove that the pairwise ranking of two samples x_i and x_j is stable with α if α is lower than a bound. The pairwise ranking can be represented as $\mathbf{F}^*_{i2} - \mathbf{F}^*_{j2}$.

We start by showing that $\mathbf{F}^*_{i2} - \mathbf{F}^*_{j2}$ is a function of entries of the normalized similarity matrix $\mathbf{S}$ and the initial state $\mathbf{Y}$.

Let $\mathbf{K} = (\mathbf{I} - \alpha\mathbf{S})^{-1}$ and $\mathbf{M} = \mathbf{I} - \alpha\mathbf{S}$. Remember that $\mathbf{S} = \mathbf{D}^{-1}\mathbf{W}$. $\mathbf{F}^*$ can be expressed as:

$$\mathbf{F}^* = (1 - \alpha)\mathbf{K}\mathbf{Y} \tag{3.13}$$

For sample x_i, its corresponding entry in $\mathbf{F}^*$ is F^*_{i2}. Since $F^*_{i1} + F^*_{i2} = 1$, ranking F^*_{i1} is equivalent to ranking F^*_{i2}. We consider only F^*_{i2}. Since we only consider the difference of F^*_{i2} and F^*_{j2}, $1 - \alpha$ can be omitted. $\mathbf{Y}_{i2} = 0, (1 \leq i \leq l)$ for the first l labeled samples and $\mathbf{Y}_{i2} = \frac{1}{2}, (l + 1 \leq i \leq n)$ for rest of the data. For sample i:

$$F^*_{i2} = \sum_{z=1}^{n} K_{iz} Y_{z2} = \frac{1}{2} \sum_{z=l+1}^{n} K_{iz} \tag{3.14}$$

Theorem 1. *K_{iz} can be expressed as:*

$$K_{iz} = \frac{1}{|\mathbf{M}|} [\sum_{x=1, x\neq z}^{n} M_{jx} \overline{M}_{ij,xz} U(j,x)](-1)^{i+z} \tag{3.15}$$

where $\overline{M}_{ij,xz}$ is the determinant of $\mathbf{M}$ when rows i, j and columns x, z are crossed out, and

$$U(j,x) \;\; = \;\; \begin{cases} (-1)^{j+x-1} & \text{if } x < z \\ (-1)^{j+x} & \text{if } x > z \end{cases} \tag{3.16}$$

Proof.

$$K_{iz} = (\mathbf{M}^{-1})_{iz} = \frac{1}{|\mathbf{M}|} \overline{M}_{iz} (-1)^{i+z} \tag{3.17}$$

where $\overline{M}_{iz}$ is the determinant of $\mathbf{M}$ when row i and column z are crossed out, and $\overline{M}_{iz}(-1)^{i+z}$ is the matrix cofactor.

By Laplace's formula, we express $\overline{M}_{iz}$ along the jth row in terms of its minors. Then $\overline{M}_{iz}$ can be expressed as:

$$\overline{M}_{iz} = \sum_{x=1, x\neq z}^{n} M_{jx} \overline{M}_{ij,xz} \begin{cases} (-1)^{j+x-1} & \text{if } x < z \\ (-1)^{j+x} & \text{if } x > z \end{cases} \tag{3.18}$$

Plugging (3.18) into (3.17), we obtain (3.15). Similarly, if we express $\overline{M}_{jz}$ along the ith row in terms of its minors, a similar expression for K_{jz} can be derived. Thus, Theorem 1 is proved. □

Given the expression for K_{iz} in (3.15), The difference between K_{iz} and K_{jz} is:

$$K_{jz} - K_{iz} = \begin{cases} \frac{1}{|\mathbf{M}|}[\sum_{x=1,x\neq z}^{n} \overline{M}_{ij,xz} \\ (M_{ix}U(i,x) - M_{jx}U(j,x))](-1)^{j+z} \\ \text{if } j-i \text{ is even} \\ \frac{1}{|\mathbf{M}|}[\sum_{x=1,x\neq z}^{n} \overline{M}_{ij,xz} \\ (M_{ix}U(i,x) + M_{jx}U(j,x))](-1)^{j+z} \\ \text{if } j-i \text{ is odd} \end{cases} \tag{3.19}$$

Therefore, the difference of F_{j2}^{*} and F_{i2}^{*} is:

$$\begin{aligned} F_{j2}^{*} - F_{i2}^{*} &= \frac{1}{2}\sum_{z=l+1}^{n}(K_{jz} - K_{iz}) \\ &= \begin{cases} \sum_{z=l+1}^{n} \frac{1}{2|\mathbf{M}|}[\sum_{x=1,x\neq z}^{n} \overline{M}_{ij,xz} \\ (M_{ix}U(i,x) - M_{jx}U(j,x))](-1)^{j+z} \\ \text{if } j-i \text{ is even} \\ \sum_{z=l+1}^{n} \frac{1}{2|\mathbf{M}|}[\sum_{x=1,x\neq z}^{n} \overline{M}_{ij,xz} \\ (M_{ix}U(i,x) + M_{jx}U(j,x))](-1)^{j+z} \\ \text{if } j-i \text{ is odd} \end{cases} \end{aligned} \tag{3.20}$$

Thus, $F^*_{j2} - F^*_{i2}$ can be expressed as a function of entries in $\mathbf{M}$ and its cofactors. Then in our analysis we separate the terms in (3.20) into two terms. First we introduce the **difference term**:

$$\begin{cases} M_{ix}U(i,x) - M_{jx}U(j,x) & j-i \text{ is even} \\ \text{or} & \\ M_{ix}U(i,x) + M_{jx}U(j,x), & j-i \text{ is odd} \end{cases} \tag{3.21}$$

and separately we consider the **cofactor term**: $\frac{\overline{M}_{ij,xz}}{|\mathbf{M}|}$. Each of these terms are analyzed separately.

3.4.2.2 Difference Term

Since $\mathbf{M} = \mathbf{I} - \alpha\mathbf{S}$, we can express M_{ix} in terms of $\mathbf{S}$.

$$\begin{cases} M_{ix} = -\alpha S_{ix} & \text{if } i \neq x \\ M_{ix} = 1 - \alpha S_{ix} & \text{if } i = x \end{cases} \tag{3.22}$$

There are four cases of (3.21):

Case 1: $i \neq x$ and $j \neq x$:

$$\begin{cases} M_{ix}U(i,x) - M_{jx}U(j,x)) = \pm\alpha(S_{jx} - S_{ix}), & \\ \qquad\qquad\qquad\qquad\qquad\qquad j-i \text{ is even} & \\ \text{or} & \\ M_{ix}U(i,x) + M_{jx}U(j,x)) = \pm\alpha(S_{jx} + S_{ix}), & \\ \qquad\qquad\qquad\qquad\qquad\qquad j-i \text{ is odd} & \end{cases} \tag{3.23}$$

The $\pm$ comes from $U(i,x)$ and $U(j,x)$.

Case 2: $i \neq x$ and $j = x$:

$$\begin{cases} M_{ix}U(i,x) - M_{jx}U(j,x)) = \pm[\alpha(S_{jx} - S_{ix}) - 1], \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is even} \\ \qquad\quad \text{or} \\ M_{ix}U(i,x) + M_{jx}U(j,x)) = \pm[1 - \alpha(S_{jx} + S_{ix})], \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is odd} \end{cases} \tag{3.24}$$

In case of small $\alpha \approx 0$,

$$\begin{cases} M_{ix}U(i,x) - M_{jx}U(j,x)) \approx \pm 1, \qquad j - i \text{ is even} \\ \qquad\quad \text{or} \\ M_{ix}U(i,x) + M_{jx}U(j,x)) \approx \pm 1, \qquad j - i \text{ is odd} \end{cases} \tag{3.25}$$

Case 3: $i = x$ and $j \neq x$:

$$\begin{cases} M_{ix}U(i,x) - M_{jx}U(j,x)) = \pm[\alpha(S_{jx} - S_{ix}) - 1], \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is even} \\ \qquad\quad \text{or} \\ M_{ix}U(i,x) + M_{jx}U(j,x)) = \pm[1 - \alpha(S_{jx} + S_{ix})], \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is odd} \end{cases} \tag{3.26}$$

The approximation of case 3 is the same as in (3.25).

Case 4: $i = x$ and $j = x$:

$$\begin{cases} M_{ix}U(i,x) - M_{jx}U(j,x)) = \pm\alpha(S_{jx} - S_{ix}), \\ \qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is even} \\ \text{or} \\ M_{ix}U(i,x) + M_{jx}U(j,x)) = \pm[2 - \alpha(S_{jx} + S_{ix})], \\ \qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is odd} \end{cases} \tag{3.27}$$

In case of small $\alpha \approx 0$,

$$\begin{cases} M_{ix}U(i,x) - M_{jx}U(j,x)) = \pm\alpha(S_{jx} - S_{ix}), \\ \qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is even} \\ \text{or} \\ \qquad\qquad\qquad M_{ix}U(i,x) + M_{jx}U(j,x)) \approx \pm 2, \\ \qquad\qquad\qquad\qquad\qquad\qquad j - i \text{ is odd} \end{cases} \tag{3.28}$$

With the above approximations, we conclude that the difference term can be either a constant value or a product of a term α and a term which is a function of $\mathbf{S}$.

3.4.2.3 Cofactor Term

We have defined $\mathbf{L}_s = \mathbf{I} - \mathbf{S}$, $\beta = \frac{1-\alpha}{\alpha}$ and $\mathbf{M} = \alpha(\beta\mathbf{I} + \mathbf{L}_s)$. Since we are only concerned about the sign of $\mathbf{M}$, the constant parameter α is dropped. The eigenvalue decomposition of $\mathbf{M}$ is $\mathbf{M} = \mathbf{V}(\mathbf{\Lambda} + \beta\mathbf{I})\mathbf{V}^{-1}$, where the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{L}_s$, and the diagonal of $\mathbf{\Lambda}$ are the eigenvalues of $\mathbf{L}_s$. $\mathbf{V}^{-1}$

is the inverse of $\mathbf{V}$. Since $\mathbf{M}$ is a symmetric matrix, $\mathbf{V}$ is invertible. Note that $\mathbf{V}$, $\mathbf{V}^{-1}$ and $\mathbf{\Lambda}$ do not depend on α. We can also write $\mathbf{M}$ as follows:

$$\mathbf{M} = \sum_{k=1}^{n} (\lambda_k + \beta)\mathbf{v}_k \mathbf{v}_k^{-1} \tag{3.29}$$

where $\mathbf{v}_k$ is the kth column of $\mathbf{V}$, and $\mathbf{v}_k^{-1}$ is kth row of $\mathbf{V}^{-1}$. By Perron-Frobenius Theorem, $|\lambda_k| \leq 2, (1 \leq k \leq n)$. If β is large enough (α is small enough), $\mathbf{M}$ is approximated by:

$$\mathbf{M} \approx \sum_{k=1}^{n} \beta \mathbf{v}_k \mathbf{v}_k^{-1} \tag{3.30}$$

Therefore, the determinant of $\mathbf{M}$ can be approximated by $|\mathbf{M}| \approx |\sum_{k=1}^{n} \beta \mathbf{v}_k \mathbf{v}_k^{-1}|$.

Similarly, $\overline{M}_{ij,xz}$ can be expressed as:

$$\overline{M}_{ij,xz} = |\sum_{k=1}^{n} (\lambda_k + \beta)\mathbf{v}_{k(-ij)} \mathbf{v}_{k(-xz)}^{-1}| \tag{3.31}$$

where $\mathbf{v}_{k(-ij)}$ is kth column of $\mathbf{V}$ with ith and jth rows crossed out, and $\mathbf{v}_{k(-xz)}^{-1}$ is kth row of $\mathbf{V}^{-1}$ with xth and zth columns crossed out. If β is large enough, $\overline{M}_{ij,xz}$ is approximated by:

$$\begin{aligned} \overline{M}_{ij,xz} &= |\sum_{k=1}^{n} (\lambda_k + \beta)\mathbf{v}_{k(-ij)} \mathbf{v}_{k(-xz)}^{-1}| \\ &\approx |\sum_{k=1}^{n} \beta \mathbf{v}_{k(-ij)} \mathbf{v}_{k(-xz)}^{-1}| \end{aligned} \tag{3.32}$$

Therefore, the cofactor term $\frac{\overline{M}_{ij,xz}}{|\mathbf{M}|}$ can be approximated by:

$$\frac{\overline{M}_{ij,xz}}{|\mathbf{M}|} \approx \frac{|\sum_{k=1}^{n} \mathbf{v}_{k(-ij)} \mathbf{v}_{k(-xz)}^{-1}|}{|\sum_{k=1}^{n} \mathbf{v}_k \mathbf{v}_k^{-1}|} \tag{3.33}$$

3.4.2.4 Approximation of Ranking

If we plug the approximation of both difference term and cofactor term into (3.19), $K_{jz} - K_{iz}$ can be approximated as follows.

Case 1: $i \neq x$ and $j \neq x$. We plug (3.23) and (3.33) into (3.19):

$$K_{jz} - K_{iz} \approx \begin{cases} \alpha(-1)^{j+z}[\sum_{x=1,x\neq z}^{n} \\ \frac{|\sum_{k=1}^{n} \mathbf{v}_{k(-ij)}\mathbf{v}_{k(-xz)}^{-1}|}{|\sum_{k=1}^{n} \mathbf{v}_k\mathbf{v}_k^{-1}|}(\pm(S_{jx} - S_{ix}))] \\ \text{if } j-i \text{ is even} \\ \alpha(-1)^{j+z}[\sum_{x=1,x\neq z}^{n} \\ \frac{|\sum_{k=1}^{n} \mathbf{v}_{k(-ij)}\mathbf{v}_{k(-xz)}^{-1}|}{|\sum_{k=1}^{n} \mathbf{v}_k\mathbf{v}_k^{-1}|}(\pm(S_{jx} + S_{ix}))] \\ \text{if } j-i \text{ is odd} \end{cases} \tag{3.34}$$

Case 2: $i \neq x$ and $j = x$. We plug (3.24) and (3.33) into (3.19):

$$K_{jz} - K_{iz} \approx \begin{cases} (-1)^{j+z}[\sum_{x=1,x\neq z}^{n} \frac{|\sum_{k=1}^{n} \mathbf{v}_{k(-ij)}\mathbf{v}_{k(-xz)}^{-1}|}{|\sum_{k=1}^{n} \mathbf{v}_k\mathbf{v}_k^{-1}|}(\pm 1)] \\ \text{if } j-i \text{ is even} \\ (-1)^{j+z}[\sum_{x=1,x\neq z}^{n} \frac{|\sum_{k=1}^{n} \mathbf{v}_{k(-ij)}\mathbf{v}_{k(-xz)}^{-1}|}{|\sum_{k=1}^{n} \mathbf{v}_k\mathbf{v}_k^{-1}|}(\pm 1)] \\ \text{if } j-i \text{ is odd} \end{cases} \tag{3.35}$$

Case 3: $i = x$ and $j \neq x$. The approximation of case 3 is the same as in (3.35).

Case 4: $i = x$ and $j = x$. We plug (3.27) and (3.33) into (3.19):

$$K_{jz} - K_{iz} \approx \begin{cases} \alpha(-1)^{j+z}[\sum_{x=1,x\neq z}^{n} \\ \frac{|\sum_{k=1}^{n} \mathbf{v}_{k(-ij)}\mathbf{v}_{k(-xz)}^{-1}|}{|\sum_{k=1}^{n} \mathbf{v}_k\mathbf{v}_k^{-1}|}(\pm(S_{jx} - S_{ix}))] \\ \text{if } j-i \text{ is even} \\ 2(-1)^{j+z}[\sum_{x=1,x\neq z}^{n} \frac{|\sum_{k=1}^{n} \mathbf{v}_{k(-ij)}\mathbf{v}_{k(-xz)}^{-1}|}{|\sum_{k=1}^{n} \mathbf{v}_k\mathbf{v}_k^{-1}|}(\pm 1)] \\ \text{if } j-i \text{ is odd} \end{cases} \tag{3.36}$$

The above equations show that the approximation of $K_{jz} - K_{iz}$ is separated into two parts. The first part is either α or a constant value. The second part is a complex term which is a function of the terms in $\mathbf{S}$. The complex term is independent of α. Since $F^*_{j2} - F^*_{i2} = \frac{1}{2}\sum_{z=l+1}^{n}(K_{jz} - K_{iz})$, $F^*_{j2} - F^*_{i2}$ can be approximated in a similar way, separated in to a term dependent on α (or a constant value) and a function of the terms in $\mathbf{S}$. Therefore, if α is small enough (below a bound), the pairwise difference remains stable with changes of α (changes below the bound).

3.4.3 Choice of the Bound

To derive the approximated pairwise ranking, two approximations are used in the cofactor term, in (3.30) and (3.32), and another two approximations are used in the difference term, in (3.25) and (3.28).

In order for the approximation in (3.30) and (3.32) to hold, β must be large enough (corresponding to small α). $\mathbf{D}^{-1}\mathbf{S}$ is a stochastic matrix. By the Perron-Frobenius theorem, the absolute values of the eigenvalues $\lambda_k, (1 \leq k \leq n)$ of $\mathbf{L}_s = \mathbf{I} - \mathbf{D}^{-1}\mathbf{S}$ are bounded by 2. For (3.30) and (3.32) to hold, it is required β is substantially greater than $|\lambda_k|, (1 \leq k \leq n)$. For example choosing bound to be $\alpha < 0.005$ will be sufficiently good bound.

In order for the approximation in (3.25) and (3.28) to hold, we assume $\mid \alpha(S_{jx} - S_{ix}) \mid < 0.01$. In this case, $\alpha < \frac{0.01}{|S_{jx} - S_{ix}|}$. The bound depends on the values in the similarity matrix. If the edge weights from a node x differ drastically, a smaller bound on α is obtained.

In summary, the bound $\mathcal{B}_\alpha$ is

$$\mathcal{B}_\alpha = \min_{i,j,x}(0.005, \frac{0.01}{\mid S_{jx} - S_{ix} \mid}) \tag{3.37}$$

3.4.4 Extension to Multi-class Ranking

If case of multi-class ranking $c > 2$, assume the labeled class is class 1. F^*_{i1} and F^*_{j1} are:

$$\begin{aligned} F^*_{i1} &= \sum_{z=1}^{l} K_{iz} + \frac{1}{c}\sum_{z=l+1}^{n} K_{iz} \\ F^*_{j1} &= \sum_{z=1}^{l} K_{jz} + \frac{1}{c}\sum_{z=l+1}^{n} K_{jz} \end{aligned} \tag{3.38}$$

$K_{jz} - K_{iz}$ is the same as in (3.19). The difference of F^*_{j1} and F^*_{i1} the becomes:

$$F^*_{j1} - F^*_{i1} = \sum_{z=1}^{l}(K_{jz} - K_{iz}) + \frac{1}{c}\sum_{z=l+1}^{n}(K_{jz} - K_{iz}) \tag{3.39}$$

$\frac{1}{c}\sum_{z=l+1}^{n}(K_{jz} - K_{iz})$ is the same as (3.20) except for a constant $\frac{1}{c}$. $\sum_{z=1}^{l}(K_{jz} - K_{iz})$ is

$$\sum_{z=1}^{l}(K_{jz} - K_{iz}) = \begin{cases} \sum_{z=1}^{l} \frac{(-1)^{j+z}}{|\mathbf{M}|}[\sum_{x=1,x\neq z}^{n} \overline{M}_{ij,xz} \\ \qquad (M_{ix}U(i,x) - M_{jx}U(j,x))] \\ \qquad\qquad \text{if } j-i \text{ is even} \\ \sum_{z=1}^{l} \frac{(-1)^{j+z}}{|\mathbf{M}|}[\sum_{x=1,x\neq z}^{n} \overline{M}_{ij,xz} \\ \qquad (M_{ix}U(i,x) + M_{jx}U(j,x))] \\ \qquad\qquad \text{if } j-i \text{ is odd} \end{cases} \tag{3.40}$$

The only difference between (3.40) and (3.20) is that the summation is from 1 to l. We can also decompose (3.40) into the product of a term dependent on α (or a constant value) and a function of items in the similarity matrix as in

(3.31),(3.33),(3.34). In multi-class case, the pairwise ranking is also stable with small α.

3.4.5 Evaluation of α in LGC Ranking

If the data points are ranked by LGC, there are two ways to retrieve the data according to the kth column of $\mathbf{F}^*$: (a) Retrieve the top ranking data points which correspond to the data points most relevant to the labeled data. (b) Retrieve the bottom ranking data points which correspond to the data points most irrelevant to the labeled data (i.e., most similar to classes other than the labeled one). In the following experiments in order to test LGC ranking, we take the second view of retrieving ranked data. Define N_R as the number of samples retrieved that belong to classes other than the labeled class. We use the following metric accuracy $\mathcal{A}$ to measure the effectiveness of LGC ranking:

$$\mathcal{A} = \frac{N_R}{R} \tag{3.41}$$

where R is the number of samples retrieved.

We first run LGC ranking on the two moon data. 20% of the left upper moon are selected as labeled data. The RBF kernel is used in calculating the similarity matrix. The bound $\mathcal{B}_\alpha$ for two moon data is calculated by (3.37), and is found to be $\mathcal{B}_\alpha = 0.005$.

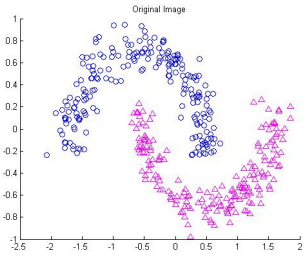
Let α take values in the set $10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 0.2, \ldots, 0.9$. Then we select R data points that are least relevant to the labeled data. In this experiment with two moon data, R is equal to the number of labeled samples. If labeled data belongs to class 1, then the data points least relevant to the labeled data are the ones with the smallest F^*_{i1} (highest F^*_{i2}). Figure 3.3 shows the R retrieved data points

for $\alpha = 0.9, 0.5, 0.1, 10^{-2}, 10^{-4}, 10^{-5}$. The blue circles are unlabeled positive data points and the green circles are labeled ones. The cyan triangles and black stars are retrieved data points. The black stars represent the retrieved data points that do not appear in the next subfigure. For example, the black stars in Figure 3.3(b) are retrieved data points that are not retrieved in Figure 3.3(c).

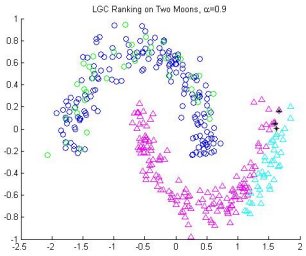
It can be observed that when α changes from 0.9 to 0.01, some of the retrieved data points change, as manifested by the black stars. The number of black stars is decreasing as α is decreasing, which means that the pairwise rankings of data points have fewer and fewer changes. When α is small, and changes from 0.1 to 10^{-5}, there are no black stars and the retrieved data remain the same. In all cases of α, $\mathcal{A}$ remains 1, which means the accuracy of retrieving data is not sensitive to α. However, the retrieved data points are not the same when α is greater than 0.1, but remain unchanged when α is lower than 0.1. The experimental results are consistent with the theoretical analysis that as long as α is small, the pairwise rankings are stable as α changes.

We also test LGC ranking on the real data sets of USPS and Mnist. Each experiment is performed on the two subsets of the data. One of the subsets is denoted as positive and other is negative. 20% of the positive class are labeled. We set R, the total number of retrieved nodes, as the total number of negative data. $\mathcal{A}$ is a measure of how many true negative are retrieved. The accuracy $\mathcal{A}$ of LGC ranking in USPS and Mnist data sets are shown in Table 3.1 and Table 3.2.

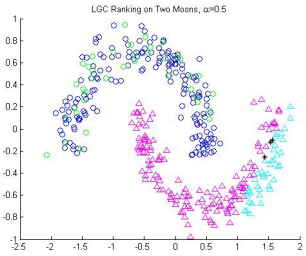
RBF kernel is also used in the similarity matrix of USPS and Mnist data sets. The bound of α for the two data sets is $\mathcal{B}_\alpha = 0.005$. We calculate the average retrieval accuracy of $\alpha > \mathcal{B}_\alpha$ and $\alpha < \mathcal{B}_\alpha$ and their standard deviations. They are shown in Table 3.3 and Table 3.4.



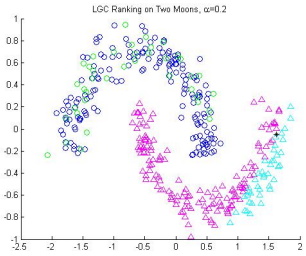
(a) Original Image



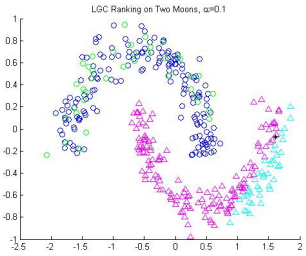
(b) $\alpha = 0.9$



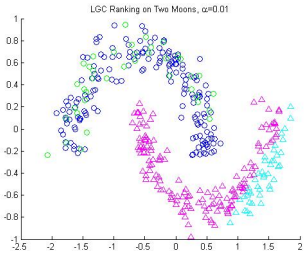
(c) $\alpha = 0.5$



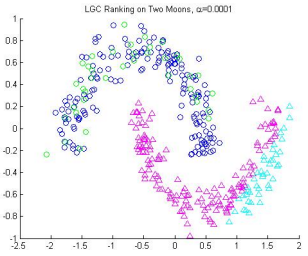
(d) $\alpha = 0.2$



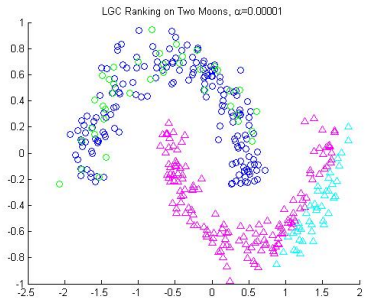
(e) $\alpha = 0.1$



(f) $\alpha = 0.01$



(g) $\alpha = 0.0001$



(h) $\alpha = 0.00001$

Figure 3.3: Two Moons data and the identified negative data in the right lower moon. The blue circles are unlabeled positive data points and the green circles are labeled ones. The cyan triangles and black stars are retrieved data points. The black stars represent the retrieved data points that do not appear in the next subfigure.

We observe that the accuracy of $\alpha < \mathcal{B}_\alpha$ is consistently higher than $\alpha > \mathcal{B}_\alpha$, and the standard deviation of $\alpha < \mathcal{B}_\alpha$ is consistently lower than $\alpha > \mathcal{B}_\alpha$ in all data sets. This is also consistent with the theoretical analysis.

Table 3.1: Accuracy of USPS with respect to α (%)

α	$\mathcal{A}$ (Class 4 and 7)	$\mathcal{A}$ (Class 0 and 1)	$\mathcal{A}$ (Class 5 and 9)	$\mathcal{A}$ (Class 2 and 8)
0.9	94.00	91.27	97.45	98.36
0.7	93.00	93.55	96.91	98.55
0.5	91.73	93.64	96.91	98.91
0.3	96.00	92.18	96.91	98.73
0.1	96.36	90.09	97.00	98.55
0.01	94.91	95.36	98.27	98.91
10^{-3}	95.55	95.82	97.27	98.45
10^{-4}	97.00	93.09	98.45	99.81
10^{-5}	95.55	97.82	97.82	99.18
10^{-6}	95.55	95.55	97.82	99.27
10^{-7}	95.82	97.18	97.91	99.45
10^{-8}	95.73	96.64	98.00	99.00

3.4.6 Summary

As the solution to the LGC algorithm, $\mathbf{F}^*$ is a function of parameter $\alpha, (0 < \alpha < 1)$. When LGC is applied to ranking, we have shown that the ranking of two data points can be decomposed into two parts. By approximations of both parts, we conclude that in case of small α, the pairwise ranking can be approximated by a product of α and a function of the similarity matrix. Thus, LGC ranking is stable when α is small. But notice that small α is only a sufficient condition that LGC ranking is independent of α, which means that in some cases a larger α may be chosen and also have little influence on LGC ranking.

Table 3.2: Accuracy of Mnist with respect to α (%)

α	$\mathcal{A}$ (Class 4 and 7)	$\mathcal{A}$ (Class 0 and 1)	$\mathcal{A}$ (Class 5 and 9)	$\mathcal{A}$ (Class 2 and 8)
0.9	97.49	100	97.54	97.67
0.7	97.81	100	97.43	97.86
0.5	98.02	100	97.64	98.15
0.3	98.02	100	97.54	97.67
0.1	97.91	100	97.64	98.05
0.01	98.23	100	96.71	96.69
10^{-3}	97.91	100	97.64	97.57
10^{-4}	98.02	100	97.33	98.35
10^{-5}	98.12	100	97.74	98.05
10^{-6}	98.23	100	97.33	98.35
10^{-7}	98.12	100	97.64	97.86
10^{-8}	98.12	100	97.43	97.76

Table 3.3: Average Accuracy and Standard Deviation of USPS $> \mathcal{B}_\alpha$ and $< \mathcal{B}_\alpha$ (mean % (std))

α	$> \mathcal{B}_\alpha$	$< \mathcal{B}_\alpha$
A (Class 4 and 7)	94.33 (1.78)	95.86 (0.57)
A (Class 0 and 1)	92.69 (1.89)	96.02 (1.66)
A (Class 5 and 9)	97.24 (0.55)	97.88 (0.38)
A (Class 2 and 8)	98.67 (0.22)	99.26 (0.18)

Table 3.4: Average Accuracy and Standard Deviation of Mnist $> \mathcal{B}_\alpha$ and $< \mathcal{B}_\alpha$ (mean % (std))

α	$> \mathcal{B}_\alpha$	$< \mathcal{B}_\alpha$
A (Class 4 and 7)	97.91 (0.25)	98.09 (0.11)
A (Class 0 and 1)	100 (0)	100 (0)
A (Class 5 and 9)	97.42 (0.35)	97.52 (0.18)
A (Class 2 and 8)	97.71 (0.49)	97.88 (0.28)

3.5 Application

In this section, we apply LGC ranking to binary one class learning. Normally, binary supervised learning problems are based on training using both labeled positive and negative data. One class learning is a kind of information retrieval using only labeled positive data. The key feature of this problem is that there are no labeled negative samples.

In recent years, there have been some algorithms proposed to solve the one class learning problem in a two step strategy, namely,

Identification, i.e., identifying a number of reliable negative samples from the unlabeled set.

Classification, i.e., building a classifier with the positive labeled samples and the selected negative ones.

The two step strategy has the advantage that there are many available conventional supervised learning tools. As long as the negative samples are accurately identified, good classification can be achieved by a state-of-the-art classifier.

The S-EM technique [33], PEBL [56] and Roc-SVM [32] are all two-step algorithms. S-EM uses Naive Bayesian, PEBL uses a positive feature set, and Roc-SVM uses a Rocchio classifier to identify negative samples. Compared with the above methods, the graph-based method can explore the connection between the samples through the paths in the graph, so the results take into account the geometric structure of the data. For the second step, S-EM uses Expectation-Maximization (EM) algorithm with Naive Bayesian (NB) classifier. PEBL and Roc-SVM both use SVM.

We propose a new two step algorithm for one class learning, which we call Graph-OCL [45]. The first step, identifying R reliable negative samples is based on LGC

ranking. We first build a graph with each node as a sample vector and edge weight as pairwise similarity. Then we select reliable negative samples by LGC ranking. Based on our analysis of the LGC ranking in the previous sections, the ranking is accurate and insensitive to the parameter α as long as α is below a certain bound ($\alpha < \mathcal{B}_\alpha$). LGC provides a simple approach to identify negative samples, since we do not have to tune any parameters by time consuming validation methods. An important issue in Identification is to determine the number of reliable negative samples R. We employ the spy sample method introduced in [33] to determine R.

For classification, we use TSVM [23] to classify the data. As we are given the unlabeled samples at the time of training and are only interested in the classification of the observed data (no out-of-sample problem), we use TSVM instead of regular SVM to build a classifier. The effectiveness of TSVM depends on the choice of kernel function. We show that in order for the data representation to be consistent in two steps, the similarity matrix used in the identification step should be a normalized kernel matrix in TSVM. We first determine the kernel matrix used in TSVM and then derive the similarity matrix from it.

We apply the complete algorithm to the 20 Newsgroup data and compare it to the Naive Bayesian (NB) [35] and S-EM methods [33]. We test the algorithm on several popular kernel functions (linear, RBF, polynomial, etc.). We found in our experiments that linear kernel matrix, corresponding to the cosine similarity matrix in Identification, has the best average classification accuracy. The improved classification accuracy show the effectiveness of Graph-OCL.

3.5.1 Identify Negative Samples

In binary one class learning, $\mathbf{F}(t), (t \geq 0)$, $\mathbf{F}^*$ and $\mathbf{Y}$ are all $n \times 2$ matrices. Assume class 1 has labeled data, and class 2 does not have any labeled data. $Y_{i1} = 1$ and $Y_{i2} = 0$ if $\mathbf{x}_i$ is labeled. $Y_{i1} = Y_{i2} = 1/2$ if $\mathbf{x}_i$ is unlabeled.

The approach for identifying negative samples by LGC ranking is shown in Algorithm 5:

Algorithm 5 Identify negative samples

1. Form the similarity matrix $\mathbf{W}$.
2. Construct matrix $\mathbf{S} = \mathbf{D}^{-1}\mathbf{W}$ in which $\mathbf{D}$ is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of $\mathbf{W}$.
3. Calculate $\mathbf{F}^* = (1 - \alpha)(\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{Y}$
4. Select R samples with hightest F^*_{i2} as the reliable negative samples.

F^*_{ij} $(j = 1, 2)$ is the probability of sample $\mathbf{x}_i$ having label j after infinite time from the initial state. The second column of $\mathbf{F}^*$ is the probability of samples belonging to the negative set given the labeled information. Graph-OCL chooses R unlabeled samples with highest F^*_{i2} as the most reliable negative samples.

3.5.2 Selecting the Number of Reliable Negative Samples

The number of negative samples R in Algorithm 5 is yet to be determined. We modified the spy document technique in S-EM [33] to decide the number of reliable negative samples. We first randomly select a subset S_R of positive samples from the labeled set and put them in the unlabeled set. Samples in S_R acts as 'spies'. The spy samples act similarly to the unknown positive samples. Hence, they allow the algorithm to infer the behavior of the unknown positive samples in the mixed set. We run Algorithm 4, then select those samples whose F^*_{i2} is greater than the highest F^*_{i2} of the spy samples. This implies that the reliable negative samples should have

a probability of belonging to negative class higher than the positive spies. However, if the spy samples happen to be close to the labeled ones in the feature space, this method will lead to more than enough negative samples, sometimes introduce false ones. So we restrict the number R to be no greater than the number of the positive labeled ones, which is denoted as l.

In summary, we first select R unlabeled samples whose F_{i2}^* are greater than the highest F_{i2}^* of the spy samples. If R is less than the number of positive labeled samples, R reliable negative samples are kept. Otherwise, only the l samples with the highest F_{i2}^* values are kept as reliable negative.

In the experiments of both toy example and real data, we set the ratio of spy samples in the labeled set to be 10%.

3.5.3 Classification by TSVM

Unlike S-EM, which uses NB based methods for both Identification and Classification, we use different methods for two steps. As mentioned in Section 3.3, LGC is not an accurate classifier, especially when the labeled data in positive and negative classes are unbalanced. Since we cannot guarantee the balance of labeled data in both classes, we use SVM based method, which is not so sensitive to the unbalance problem. We use Transductive SVM (TSVM) as a classifier, which takes into consideration of both labeled and unlabeled data during training.

Kernel Selection

Though we use different methods in Identification and Classification, these two methods are not completely separate. They are related through the similarity matrix in Identification and the kernel matrix of TSVM in Classification. The

choice of both matrices are important, because they reflect the structure of the data.

The kernel function $K(x_i, x_j)$ is the inner product of the higher dimensional mapping of the original samples x_i and x_j. If no mapping to higher dimension is performed, the kernel is the inner product of original vectors, which corresponds to linear kernel.

In order to exploit the data structure in Identification and Classification in a consistent way, we apply the same kernel to both steps. We use a normalized kernel matrix $\mathbf{K_n}$ as the similarity matrix $\mathbf{W}$ when identifying negative samples, and the original kernel matrix $\mathbf{K}$ in TSVM. The kernel matrix is transformed to a normalized one as:

$$\mathbf{K_n}(x_i, x_j) = \frac{\mathbf{K}(x_i, x_j)}{\sqrt{\mathbf{K}(x_i, x_i)\mathbf{K}(x_j, x_j)}} \tag{3.42}$$

In other words, this is the normalized inner product of sample data vectors. If the similarity matrix is defined as the cosine similarity, the corresponding kernel in TSVM is linear kernel. Defining either the similarity matrix or the kernel matrix first automatically decides the other matrix. To show the effectiveness of the above idea, we present the classification of using cosine similarity in Identification and RBF and polynomial kernel in Classification. We show in Table (3.5) that linear kernel provides the best classification results.

3.5.4 Graph-OCL Algorithm

The complete algorithm of Graph-OCL is described in Algorithm 6.

Algorithm 6 Graph-OCL Algorithm

1. Select S_R samples from the labeled positive set to form the spy samples and move them to the unlabeled set.
2. Select a kernel matrix **K** for TSVM and derive the similarity matrix **W** from K.
3. Run Algorithm 5 to determine R.
4. Run Algorithm 5 again and select R most reliable negative samples.
5. Run TSVM based on the labeled positive samples and selected negative samples.

3.5.5 Experimental Results

We test the algorithm on 20 Newsgroups [30]. This is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups, which are also categorized into 4 main categories, computer, recreation, science and talk, as shown in Figure 3.4.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Figure 3.4: 20 Newsgroup Data Subjects

The first step in document classification is to transform the documents, which are typically strings of characters into a representation suitable for learning algorithm. We use vector space model as a representation of documents. Suppose D is the corpus of documents. Each document d in D is represented as a feature vector $[w_1, w_2, \cdots, w_m]$, where m is the total number of words. Each distinct word corresponds to a feature, with the number of times the word occurs in the document as its value, i.e., w_j corresponds to the number that word j occurs in the document.

This leads to extremely long feature vectors. To avoid unnecessary words that are not discriminative, we first remove the 'stop-words' (like 'and', 'or', etc.). Then words are considered as features if they occur at least 3 times in the training data. For simplicity, we refer to m as the number of features after feature reduction. Finally, we scale the feature vector by their TF-IDF.

Term frequency (tf) is a measure of the importance of word j within the particular document $\mathbf{d}_i$.

$$tf_{ij} = n_{ij} / \sum_k n_{ik} \tag{3.43}$$

where n_{ij} is the number of occurrences of the considered word j in document $\mathbf{d}_i$, and the denominator is the sum of number of occurrences of all words in document $\mathbf{d}_i$.

The inverse document frequency (idf) is a measure of the general importance of a word.

$$idf_j = \log \frac{|D|}{|d : \text{word } j \in d|} \tag{3.44}$$

$|D|$ is the total number of documents in the corpus. $|d : \text{word } j \in d|$ is the number of documents where word j appears.

$$tf\text{-}idf_{ij} = tf_{ij} \times idf_j \tag{3.45}$$

With $tf\text{-}idf_{ij}$ as feature, document $\mathbf{d}_i$ is represented as a vector with jth column as $tf\text{-}idf_{ij}$. We build a weighted graph, in which each node represents a document. The cosine similarity of two documents defines edge weight between two nodes.

$$W_{ij} = \frac{\mathbf{d}_i \cdot \mathbf{d}_j}{\| \mathbf{d}_i \| \| \mathbf{d}_j \|} \tag{3.46}$$

We choose different groups in Newsgroup 20 data as positive class, then using various individual groups as unlabeled samples. For each experiment, we divide the positive class into two subsets, labeled and unlabeled. In the experiments, we set the labeled positive set to be 20% of the positive class and 10% of the labeled positive set as spy documents. For TSVM, we use the SVMlight package [22]. The bound $\mathcal{B}_\alpha$ for the 20 Newsgroup data is 0.005, and α is set to be 0.001 in the experiments.

We measure the algorithm by both classification accuracy and an information retrieval measure F score, defined as: $F = 2pr/(p + r)$, where p is the precision and r is the recall. F score measures the performance of a system on a particular class [44], and reflects an average effect of both precision and recall.

Table 3.5 presents the classification accuracy of using different kernels in TSVM with corresponding similarity matrix in Identification. It shows that linear kernel has the best performance in most data sets, so linear kernel is used in Graph-OCL to compare with other algorithms. Table 3.6 shows that Graph-OCL has better classification accuracy than NB and S-EM. Graph-OCL is also more stable, since all accuracies are above 85%, but NB and S-EM have some accuracy lower than 80%. The F-score of Graph-OCL is much higher than NB and S-EM, which means both precision and recall are good.

Table 3.5: Classification Accuracy of Documents with Different Kernels

kernel type	os-win/ wind.x	graphic/ hardmac	pol.guns/ pol.misc	hockey/ baseball
linear	89.1%	93.3%	90.7%	96.8%
RBF	88.5%	65.5%	59.4%	96.0%
polynomial	88.8%	93.1%	90.8%	96.3%

Table 3.6: Classification Accuracy of Documents (Linear Kernel)

Positive	Negative	Graph-OCL F score	Graph-OCL Accuracy	NB F score	NB Accuracy	S-EM F score	S-EM Accuracy
os-win	wind.x	88.0%	89.1%	45.6%	79.9%	92.2%	95.8%
graphic	hardmac	92.8%	93.3%	14.9%	73.6%	41.1%	78.5%
rel.misc	pol.misc	87.0%	89.0%	30.8%	75.7%	65.4%	82.7%
hockey	baseball	96.4%	96.8%	89.6%	95.4%	96.8%	98.4%
pol.guns	pol.misc	90.0%	90.7%	48.3%	79.9%	75.4%	87.3%
politics	rec	83.9%	86.2%	28.8%	72.6%	29.7%	73.1%
hardmac	os-win	88.9%	90.1%	74.2%	88.2%	94.4%	96.8%
hardpc	hardmac	83.6%	85.0%	52.8%	90.8%	82.7%	95.4%
average		**88.8%**	**90.0%**	48.1%	82.1%	72.2%	88.5%

3.6 Conclusions

We analyze the LGC algorithm for both classification and ranking. We propose an information diffusion interpretation of LGC, then analyze the parameter α's impact on both classification and ranking. It is found that α has a large impact on the classification results, especially when the labeled data in classes are unbalanced. For LGC ranking, we theoretically prove that the pairwise ranking remains unchanged when α is smaller than a bound. In other words, α does not have to be tuned by validation as long as it is small enough. We apply LGC ranking to one class learning as a first step to identify negative samples, and propose a new one class learning algorithm Graph-OCL. Experimental results have demonstrated the effectiveness of both LGC ranking and Graph-OCL. However, graph-based methods have a scalability problem. For large size problems (e.g., data size>10,000) both storing the similarity matrix and solving the associated linear systems are prohibitive. Approximation methods are necessary to solve large scale graph-based algorithms.

Chapter 4

A Novel Adaptive Nyström Method

4.1 Introduction

Kernel methods play an important role in machine learning and have been used successfully in algorithms such as support vector machines (SVMs) [11], kernel principal component analysis (KPCA) [43] and manifold learning [40]. One example is the use of the kernel matrix in the Graph-OCL algorithm introduced in Chapter 3.

A major challenge in applying kernel methods to modern learning comes from the fact that these usually involve large data sets of tens of thousands to millions of data points, leading to significant complexity in terms of both space (quadratic) and time (usually cubic). A similar problem may occur in graph based algorithms in which using the similarity matrix of the graph also involves significant complexity. One solution to deal with such large data sets is to use an approximation of the kernel matrix. The Nyström method [55] is an efficient technique for obtaining a low-rank approximation of a large kernel matrix, *using only a subset of its columns*. A key problem in the Nyström method, and its extensions, is that of determining which subset of the columns is to be used. The complexity and quality of the approximation depends heavily on the selected subset.

While column selection based on random sampling with a uniform probability distribution has been the most popular approach, a considerable amount of research work has been conducted exploring other sampling methods. [3] proved that if the number of independent sampled columns is equal to the rank of the kernel matrix, a perfect reconstruction can be achieved. However, finding these independent columns is difficult given the large data size. Our work develops a novel sampling strategy taking as a starting point two recently proposed methods [46]. The first approach selects columns iteratively using an adaptive sampling method, where at each step probabilities associated to each column are updated based on the quality of the intermediate approximations to the kernel matrix that are obtained [28]. A second approach makes use of greedy sampling, finding the best rank-1 Nyström approximation at each iteration and then adding the corresponding approximations to obtain an aggregate one [16]. Both algorithms are iterative; they are based on building intermediate approximations and then obtaining the final result using the set of all the selected columns.

Our proposed work is motivated by a novel interpretation of the Nyström approximation method. Assume the data is a matrix, in which rows represent samples and columns represent features. We show that sampling the columns of the kernel matrix is equivalent to projecting the data onto the subspace spanned by the corresponding columns. Based on this, a good Nyström approximation can be achieved if the space spanned by the sampled columns has a large intersection with the space spanned by the data mapped to the eigenvectors corresponding to the top eigenvalues of the kernel matrix. We verify this property using a simple experiment with a randomly generated kernel matrix. We also note that if subsets of columns exhibit this property, then it is likely that their union will also have the same property. Obviously the property cannot be used in sampling columns since it would require

knowing the top eigenvectors of the kernel matrix. However, as will be shown, it is possible to make use of alternative metrics to quantify the suitability of a column or a set of columns.

In this chapter, we proposed a novel technique for iterative, low complexity column selection. It extends the Ensemble Nyström approximation method [27] by including greedy selection of columns and adaptive probability update. Unlike [16] we do not perform a greedy column-by-column selection, which is inefficient and requires quadratic complexity. Instead, *we compare the approximation properties of several subsets of columns*, typically using the Frobenius error. The underlying assumption is that the subsets with lower error will also provide better approximation. This approach is faster than column-wise selection, since a smaller number of alternatives needs to be evaluated, i.e., the number of subsets is much smaller than the number of columns. As in [16], the approach is greedy, that is, the best subset is selected and will be included in the computation of the final estimate. However, unlike [16], each iteration is based on random sampling based on estimated probabilities for the remaining columns, which are obtained from previous iterations. We adopt the name BoostNyström, because each iteration is used to update the probability of the remaining columns. The adaptive probability update gives higher probability to potentially good columns. Note that our adaptive probability estimation differs from that of [28]. BoostNyström identifies potentially good columns as those with low approximation error in the current iteration, and thus it does not impose any requirement on the rank as is the case in [28].

In the final iterative step, Ensemble Nyström approximation is performed on the set of columns selected through the iterations. Standard Ensemble Nyström has high variance because of its use of randomly sampled columns. BoostNyström reduces the high variance by incrementally adding a subset of columns with low

approximation error at each iteration. We derive an error bound for BoostNyström, which guarantees a better convergence rate than both the standard Nyström and the Ensemble Nyström methods. Experimental results show the effectiveness of BoostNyström.

4.2 Nyström Approximation

In many machine learning algorithms, we need a matrix, each item of which represents the pairwise relationship between the two samples. The matrix can be a similarity matrix in graph based algorithms, a distance matrix or a kernel matrix in some kernel based algorithms. The similarity matrix and the kernel matrix used in Graph-OCL are two examples. As the number of samples grows, it is increasing difficult to store and calculate this type of matrix. Nyström approximation is a way to approximate the matrix by using only a few columns, such that the computational and memory complexity can be greatly reduced.

4.2.1 Standard Nyström Method

Let $\mathbf{K}$ be a symmetric positive semidefinite (SPSD) matrix. Any kernel matrix, inner product matrix or graph similarity matrix is a SPSD matrix, so we will discuss the Nyström approximation in terms of a general SPSD matrix. The Nyström approximation of $\mathbf{K}$ is obtained by sampling $m \ll n$ columns of $\mathbf{K}$. Let $\mathbf{E}$ denote the $n \times m$ matrix consisting of these sampled columns. Let $\mathbf{W}$ be the $m \times m$ matrix formed by the intersection of the m sampled columns with the corresponding m rows of $\mathbf{K}$. We write $\mathbf{K}$ and $\mathbf{E}$ in terms of their sampled submatrices as follows:

$$\mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{K}_{21}^T \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \text{ and } \mathbf{E} = \begin{bmatrix} \mathbf{W} \\ \mathbf{K}_{21} \end{bmatrix} \tag{4.1}$$

The Nyström method generates a rank-k approximation $\tilde{\mathbf{K}}$ of $\mathbf{K}$ for $k \leq m$ defined by:

$$\tilde{\mathbf{K}} = \mathbf{E}\mathbf{W}_k^+\mathbf{E}^T \approx \mathbf{K} \tag{4.2}$$

where $\mathbf{W}_k$ is the best rank-k approximation of $\mathbf{W}$ for the Frobenius norm, and $\mathbf{W}_k^+$ denotes the pseudo-inverse of $\mathbf{W}_k$. Note that $\mathbf{W}$ is also SPSD since $\mathbf{K}$ is SPSD. $\mathbf{W}_k^+$ can be derived from the singular value decomposition (SVD) of $\mathbf{W}$. $\mathbf{W} = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T$, where $\tilde{\mathbf{V}} = [\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \cdots, \tilde{\mathbf{v}}_m]$ is orthonormal and $\tilde{\mathbf{\Sigma}} = diag(\sigma_1, \cdots, \sigma_m)$ is a real diagonal matrix with $\sigma_1 \geq \cdots \geq \sigma_m \geq 0$. For $k \leq rank(\mathbf{W})$, it is given by $\mathbf{W}_k^+ = \sum_{i=1}^k \sigma_i^{-1}\tilde{\mathbf{v}}_i\tilde{\mathbf{v}}_i^T$ where $\tilde{\mathbf{v}}_i$ denotes the ith column of $\tilde{\mathbf{V}}$. Since the running time complexity of SVD is $O(m^3)$ and $O(nmk)$ is required for multiplication with $\mathbf{E}$, the total complexity of the Nyström approximation computation is $O(m^3 + nmk)$.

4.2.2 Novel Perspective

Assume that a linear kernel is used (the same proof applies to other kernel types, as any kernel matrix implicitly maps data vectors to a high-dimensional linear space.) The kernel matrix is calculated as $\mathbf{K} = \mathbf{X}^T\mathbf{X}$, where $\mathbf{X}$ is a $dim \times n$ data matrix, and dim is the number of features in each data vector. We simplify the Nyström approximation by fixing $k = m$ and assuming $\mathbf{W}$ is full rank. Then the Nyström approximation of (4.2) based on a set S of m randomly sampled columns becomes:

$$\tilde{\mathbf{K}} = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^T \tag{4.3}$$

Let $\mathbf{X}_m$ be the matrix containing the m sampled columns of $\mathbf{X}$ corresponding to the columns sampled in $\mathbf{K}$, then $\tilde{\mathbf{K}}$ can be written as:

$$\tilde{\mathbf{K}} = \mathbf{X}^T\mathbf{X}_m(\mathbf{X}_m^T\mathbf{X}_m)^{-1}\mathbf{X}_m^T\mathbf{X} \tag{4.4}$$

Since $\mathbf{X}_m^T\mathbf{X}_m$ is also an SPSD matrix, it can be decomposed in terms of its eigenvalues and eigenvectors:

$$\mathbf{X}_m^T\mathbf{X}_m = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \tag{4.5}$$

where $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m]$ is orthonormal and $\mathbf{q}_i (i = 1, 2, \cdots, m)$ is the i-th eigenvector of $\mathbf{X}_m^T\mathbf{X}_m$. $\mathbf{\Lambda}$ is the diagonal matrix containing the eigenvalues $\beta_1, \beta_2, \cdots, \beta_m$ as diagonal elements and $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_m$. The approximated kernel matrix $\tilde{\mathbf{K}}$ can then be expressed as:

$$\begin{aligned} \tilde{\mathbf{K}} &= \mathbf{X}^T\mathbf{X}_m(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T)^{-1}\mathbf{X}_m^T\mathbf{X} \\ &= \mathbf{X}^T\mathbf{X}_m(\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}) \cdot (\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^T)\mathbf{X}_m^T\mathbf{X} \end{aligned} \tag{4.6}$$

where $\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}} = [\frac{1}{\sqrt{\beta_1}}\mathbf{q}_1, \frac{1}{\sqrt{\beta_2}}\mathbf{q}_2, \cdots, \frac{1}{\sqrt{\beta_m}}\mathbf{q}_m]$ are the eigenvectors normalized by the square root of their corresponding eigenvalues.

$\mathbf{X}^T\mathbf{X}_m$ is the original data mapped onto the m sampled columns in $\mathbf{X}_m$. Then $\mathbf{X}^T\mathbf{X}_m(\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}})$ represents the mapped data $\mathbf{X}^T\mathbf{X}_m$ expressed in terms of $\mathbf{q}_i (i = 1, 2, \cdots, m)$ and normalized by $\frac{1}{\sqrt{\beta_i}}$; $\sqrt{\beta_i}$ is the magnitude of $\mathbf{X}_m$ along the vector $\mathbf{q}_i$. Thus, $\mathbf{X}^T\mathbf{X}_m(\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}})$ maps columns of $\mathbf{X}$ onto the space, denoted by SP_m, spanned by the bases $[\frac{1}{\sqrt{\beta 1}}\mathbf{X}_m\mathbf{q}_1, \frac{1}{\sqrt{\beta_2}}\mathbf{X}_m\mathbf{q}_2, \cdots, \frac{1}{\sqrt{\beta_m}}\mathbf{X}_m\mathbf{q}_m]$. Define $\mathbf{U}_m = \mathbf{X}_m(\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}})$. The ith column of $\mathbf{U}_m$ is $[\frac{1}{\sqrt{\beta i}}\mathbf{X}_m\mathbf{q}_i$ $\tilde{\mathbf{K}}$ is the linear kernel of $\mathbf{X}^T\mathbf{U}_m$. Note that SP_m

is the same space spanned by the columns of $\mathbf{X}_m$, and the columns of $\mathbf{U}_m$ are the orthonormalized bases of SP_m. Thus, $\tilde{\mathbf{K}}$ is the linear kernel of the original data projected onto the space spanned by the sampled columns, and the projection matrix is formed with the orthonormal bases obtained from the eigendecomposition of $\mathbf{X}_m^T\mathbf{X}_m$.

On the other hand, the eigendecomposition of $\mathbf{K}$ is:

$$\mathbf{K} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T \tag{4.7}$$

where $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]$ contains the eigenvectors of $\mathbf{K}$ as columns, and the diagonal values of $\mathbf{\Sigma}$ contain the eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_n$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

Since $\mathbf{X}^T\mathbf{X}\mathbf{v}_i = \lambda_i\mathbf{v}_i, (i = 1, 2, \cdots, n)$, $\mathbf{K}$ can be written as:

$$\begin{aligned}
\mathbf{K} &= \mathbf{X}^T\mathbf{X}(\frac{\mathbf{v}_1}{\sqrt{\lambda_1}}, \frac{\mathbf{v}_2}{\sqrt{\lambda_2}}, \cdots, \frac{\mathbf{v}_n}{\sqrt{\lambda_n}}) \cdot \\
&\quad (\frac{\mathbf{v}_1}{\sqrt{\lambda_1}}, \frac{\mathbf{v}_2}{\sqrt{\lambda_2}}, \cdots, \frac{\mathbf{v}_n}{\sqrt{\lambda_n}})^T\mathbf{X}^T\mathbf{X} \\
&= \mathbf{X}^T\mathbf{X}(\mathbf{V}\mathbf{\Sigma}^{-\frac{1}{2}}) \cdot (\mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{V}^T)\mathbf{X}^T\mathbf{X}
\end{aligned} \tag{4.8}$$

where (4.8) has the same structure as (4.6), and $\mathbf{U} = \mathbf{X}(\mathbf{V}\mathbf{\Sigma}^{-\frac{1}{2}})$ are the orthonormal bases of the n dimensional space. Both $\mathbf{K}$ and $\tilde{\mathbf{K}}$ can be written in a linear kernel form:

$$\begin{aligned}
\mathbf{K} &= \mathbf{X}^T\mathbf{U}\mathbf{U}^T\mathbf{X} \\
\tilde{\mathbf{K}} &= \mathbf{X}^T\mathbf{U}_m\mathbf{U}_m^T\mathbf{X}
\end{aligned} \tag{4.9}$$

The approximation error of $\tilde{\mathbf{K}}$ is:

$$
\begin{aligned}
\| \tilde{\mathbf{K}} - \mathbf{K} \| &= \| \mathbf{X}^T(\mathbf{U}_m\mathbf{U}_m^T - \mathbf{U}\mathbf{U}^T)\mathbf{X} \| \\
&= \| \mathbf{V}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{U}^T(\mathbf{U}_m\mathbf{U}_m^T - \\
&\quad \mathbf{U}\mathbf{U}^T)\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^T \| \\
&= \| \mathbf{V}\mathbf{\Sigma}^{\frac{1}{2}}(\mathbf{U}^T\mathbf{U}_m\mathbf{U}_m^T\mathbf{U} - \mathbf{I})\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{V}^T \|
\end{aligned}
\tag{4.10}
$$

According to (4.10), a sampling scheme $\mathbf{U}_m$ has 0 approximation error if $\mathbf{U}^T\mathbf{U}_m\mathbf{U}_m^T\mathbf{U} = \mathbf{I}$. Sampling all columns of $\mathbf{K}$ certainly leads to zero error. However, when the rank of $\mathbf{U}_m$ is lower than the rank of $\mathbf{X}$, it is impossible for $\mathbf{U}^T\mathbf{U}_m\mathbf{U}_m^T\mathbf{U} = \mathbf{I}$ to hold. We now describe how the corresponding error depends on properties of the chosen columns.

$\mathbf{U}_m^T\mathbf{U}$ can be expressed as:

$$
\mathbf{U}_m^T\mathbf{U} = [\mathbf{U}_m^T\mathbf{U}^1, \mathbf{U}_m^T\mathbf{U}^2, \cdots, \mathbf{U}_m^T\mathbf{U}^n] \tag{4.11}
$$

where $\mathbf{U}^i$ is the ith column of $\mathbf{U}$. For a given rank m, in order for (4.10) to be small, $\mathbf{U}^T\mathbf{U}_m\mathbf{U}_m^T\mathbf{U}$ should be close to $\mathbf{I}$. Since $(\mathbf{U}^T\mathbf{U}_m\mathbf{U}_m^T\mathbf{U} - \mathbf{I})$ is weighted by $\mathbf{\Sigma}$ in (4.10), two properties would be desirable for $\mathbf{U}_m^T\mathbf{U}$:

(a) The energy of $\mathbf{U}_m^T\mathbf{U}$ should be focused on the left part of the matrix. In other words, $\| \mathbf{U}_m^T\mathbf{U}^i \| \geq \| \mathbf{U}_m^T\mathbf{U}^j \|$, $(i \leq j)$. Intuitively this means that the subspace SP_m has more intersection with subspaces spanned by $\{\frac{\mathbf{X}\mathbf{v}_i}{\sqrt{\lambda_i}}\}(1 \leq i \leq n)$, for large λ_i.

(b) The rows of $\mathbf{U}_m^T\mathbf{U}$ are orthogonal, which means that $v_i^T\tilde{\mathbf{K}}v_j = 0, (1 \leq i, j \leq n, i \neq j)$.

In order to sample columns that satisfy Properties (a) and (b), we would need to know the eigenvectors and eigenvalues of $\mathbf{K}$ in advance. However, this is not feasible in the case of large data sizes. Instead, we use an alternative metric, the approximation error on a validation set, to measure how good the approximate kernel matrix is. A low approximation error is equivalent to having Properties (a) and (b). In various iterative algorithms for column sampling, including [28], [16] and our proposed BoostNyström, a subset of columns that are considered to provide good approximation is selected at each iterative step. The final set of selected columns is the union of the subsets at each iterative step. It is not obvious why a union of good subsets would still have good approximation properties. However, this can be verified based on Properties (a) and (b). Since a low approximation error is equivalent to having Properties (a) and (b), if it is proved that Properties (a) and (b) of a subset can be extended to its union, then the iterative algorithm is justified.

Property (a) states that in order to have low approximation error, the subspace spanned by the selected columns should have large overlap with the subspace obtained from the eigenvectors corresponding to the top eigenvalues of $\mathbf{K}$. A union of subsets of columns can be considered as a sum of subspaces. A sum of subspaces that have Property (a) certainly conforms to Property (a) too. This proves that Property (a) can be extended to union of subsets.

However, Property (b) does not hold in general when extended to a union of subsets. However, we can empirically prove that, as compared to Property (b), Property (a) is a dominant factor in determining approximation error. We propose a metric $\mathcal{M}$ to measure Property (a):

$$\mathcal{M} = \| \ (\mathbf{X}_m(\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}))^T \cdot \mathbf{X}(\mathbf{V}\mathbf{\Sigma}^{-\frac{1}{2}}) \cdot \mathbf{\Sigma}^2 \ \|_F \tag{4.12}$$

We perform an experiment on a 20×20 matrix $\mathbf{X}$. Each item of $\mathbf{X}$ is randomly generated from a uniform distribution in the interval $[-10, 10]$. Then we randomly sample 3 columns and use them to compute a Nyström approximation. We repeat the sampling process 10000 times. Among these 10000 approximations, we calculate each approximation's error ε and its $\mathcal{M}$ value. For two Nyström approximations i and j, if $\varepsilon_i < \varepsilon_j$ and $\mathcal{M}_i > \mathcal{M}_j$, or $\varepsilon_i > \varepsilon_j$ and $\mathcal{M}_i < \mathcal{M}_j$, we say the $\mathcal{M}$ is effective for the two approximations. In our experiment, we observe that over 95% of the pairs are effective.

Thus, we have empirically shown that Property (a) is a dominant factor in determining approximation error. Since Property (a) can be extended to a union of subsets, the iterative procedure of selecting subsets of columns with low approximation errors and combining them is justified. In the next section, we introduce BoostNyström based on the above result.

4.3 Proposed Method

We first introduce the Ensemble Nyström approximation method, which is used in the final iteration of BoostNyström. We then introduce our proposed BoostNyström algorithm.

4.3.1 Ensemble Nyström Method

The Ensemble Nyström algorithm [27] leads to an improved hypothesis by sampling uniformly without replacement a set S of $mp, (p > 1)$ columns from matrix $\mathbf{K}$. S is decomposed into p subsets $S_1, \cdots, S_p$. Each subset $S_r, (r \in [1, p])$ contains m columns and is used to obtain a rank-k Nyström approximation $\tilde{\mathbf{K}}_r$. $\tilde{\mathbf{K}}_r$ can be written as $\tilde{\mathbf{K}}_r = \mathbf{E}_r \mathbf{W}_r^+ \mathbf{E}_r^T$, where $\mathbf{E}_r$ and $\mathbf{W}_r$ denote the matrices formed by the

columns of S_r. The general form of the approximation of $\mathbf{K}$ generated by the ensemble Nyström algorithm is:

$$\tilde{\mathbf{K}}^{ens} = \sum_{r=1}^{p} \mu_r \tilde{\mathbf{K}}_r \tag{4.13}$$

The mixture weights μ_r can be defined in different ways. One choice is to assign equal weight to each base approximation, $\mu_r = \frac{1}{p}, (1 \leq r \leq p)$. Another choice is exponential weight method, which first measures the approximation error of the rth base approximation, $\hat{\varepsilon}_r$, and then computes the mixture weight as $\mu_r = \exp(-\eta\hat{\varepsilon}_r)/Z$, with $Z = \sum_{j=1}^{p} \mu_j$.

4.3.2 BoostNyström Method

BoostNyström is an iterative algorithm, which extends the Ensemble Nyström method by greedily selecting the best sampled subset and adaptively determining the sampling distribution of the remaining columns in each iteration.

Let T be the total number of iterations. First, a set of sampled columns S is generated through T iterations. Then the Ensemble Nyström method is applied to S.

At iterative step t, $m(p-t+1)$ columns are sampled according to the probability distribution $\mathbf{P}_t (1 \leq t \leq T)$. Initially $\mathbf{P}_1$ is a uniform distribution and a set of mp columns are sampled. The set of columns is decomposed into $\frac{mp}{d}$ subsets of size d, on which a standard Nyström method is performed. Then, the $\frac{mp}{d}$ subsets are sorted in ascending order based on their approximation errors on the validation set V. The m columns that contain the m/d subsets having the lowest approximation errors, are added to S. The next m/d subsets which have the second lowest approximation errors are given higher probability distribution in the next iteration. Then the

probability of the $\frac{m(p-1)}{d}$ columns are normalized so that the probability sum is equal to 1. At the second iterative step, we sample m fewer columns than the first iteration, and repeat the same process until iteration T. Figure 4.1 illustrates how S is generated through T iterations.

After T iterations, we run an Ensemble Nyström approximation using S, which is decomposed into p subsets of size m. Exponential mixture weights are used in the Ensemble Nyström approximation. Let $\mu_j, 1 \leq j \leq p$ be the mixture weight in (4.13) at iteration T. Define $\eta > 0$ as a parameter to calculate the mixture weights. In the algorithm, we set $p = T$.

The complete algorithm is described in Algorithm 7. The BoostNyström algorithm uses two methods

update probability and *build mixture weight*, which are described in Algorithm 8 and Algorithm 9 respectively.

Algorithm 7 BoostNyström

---

1. $t = 1$ and set the initial probability distribution $\mathbf{P}_1$ as uniform distribution.
2. Sample a set V of m columns as the validation set.
3. Sample a set of $m(p - t + 1)$ columns using the probability distribution $\mathbf{P}_t$ from matrix $\mathbf{K}$ without replacement.
4. The sampled columns are decomposed into $\frac{m(p-t+1)}{d}$ subsets $S_1, S_2, \cdots, S_{\frac{m(p-t+1)}{d}}$. Each subset $S_r(r = 1, 2, \cdots, \frac{m(p-t+1)}{d})$ contains d columns.
5. Obtain the rank-k Nyström approximation $\mathbf{K}_r$ on S_r, and calculate the approximation error ε_r of $\mathbf{K}_r$ on V.
6. Rank $\varepsilon_r(r = 1, 2, \cdots, \frac{m(p-t+1)}{d})$ ($S_{(r)}$ accordingly) in ascending order, and $S_{(r)}(\text{sorted} 1 \leq r \leq \frac{m}{d})$ into S.
7. If $t < T$, construct $\mathbf{P}_{t+1}$ based on $\varepsilon_r(r = 1, 2, \cdots, \frac{m(p-t+1)}{d})$ by using Algorithm 8;
8. If $t = T$, build an Ensemble Nyström approximation on samples S with each subset of size m and mixture weights $\mu_j(j = 1, 2, \cdots, p)$. Construct $\mu_j(j = 1, 2, \cdots, p)$ by using Algorithm 9.
9. $t = t + 1$. If $t > T$, then stop the iteration. Otherwise, go to Step 3.

Algorithm 8 *update probability*

1. Assume $\varepsilon_{(r)}, (r = 1, 2, \cdots, \frac{m(p-t+1)}{d})$ are sorted in ascending order. Derive the average approximation error $\varepsilon_{avg} = \frac{d}{m(p-t+1)} \sum_{r=1}^{\frac{m(p-t+1)}{d}} \varepsilon_{(r)}$.
2. Find $\frac{m}{d}$ subsets $S_{(r)}, (r = 1, 2, \cdots, \frac{m}{d})$ with smallest approximation errors and calculate $\varepsilon_{min} = \frac{d}{m} \sum_{r=1}^{\frac{m}{d}} \varepsilon_{(r)}$.
3. Calculate ratio $= \varepsilon_{avg}/\varepsilon_{min}$. Multiply the probability distribution of the subsets $S_{(r)}, (r = \frac{m}{d} + 1, \frac{m}{d} + 2, \cdots, \frac{2m}{d})$ by this ratio.
4. Force the probability distribution of the subsets $S_{(r)}, (r = 1, 2, \cdots, \frac{m}{d})$ to be 0.
5. Normalize $\mathbf{P}_{t+1}$ so that $\sum_{i=1}^{n} P_{t+1,i} = 1$.

Algorithm 9 *build mixture weight*

1. Calculate $\hat{\varepsilon}_j, (j = 1, 2, \cdots, p)$ as the approximation error of jth subset in S on V .
2. Define $\mu_j = \exp(-\eta\hat{\varepsilon}_j), (j = 1, 2, \cdots, p)$.
3. $Z = \sum_{j=1}^{p} \mu_j$, and normalize $\mu_j = \mu_j/Z$.

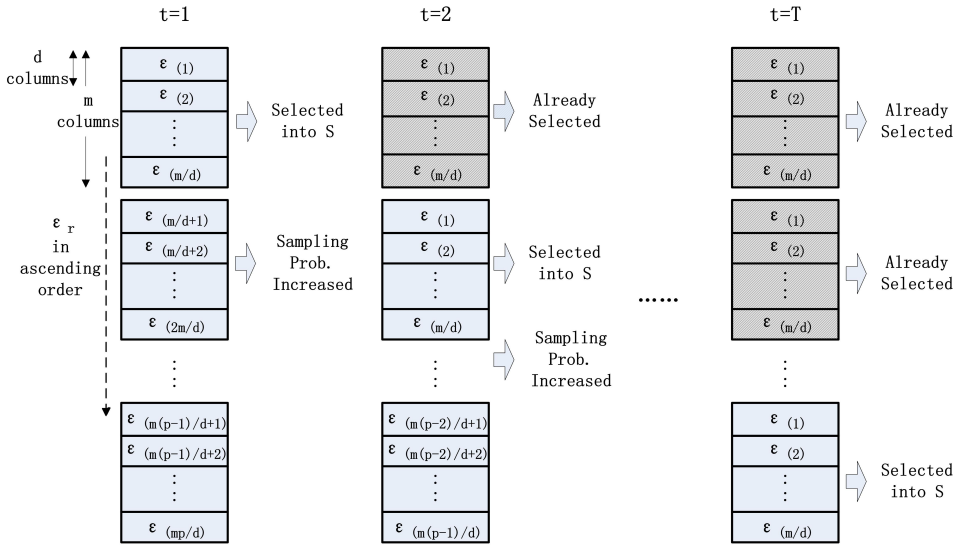


Figure 4.1: BoostNyström Algorithm

4.3.3 Complexity Analysis

The time complexity of performing a standard Nyström approximation on a size d subset is $O(d^3 + nd^2)$. Since there are at most $\frac{mp}{d}$ such subsets in each iteration,

the total complexity of performing standard Nyström approximation is $O(mpd^2 + nmpd)$. There are additional complexities including sorting the approximation errors ($C_s = O(\frac{mp}{d}\log(\frac{mp}{d}))$), updating the probability distribution ($C_u = O(n)$) and computing the Ensemble Nyström approximation at the final iterative step ($C_\mu = O(pm^3 + pm^2n)$). The time complexity of the iterations in BoostNyström is $O(T(mpd^2 + nmpd + C_s + C_u) + C_\mu)$. The dominant term is $O(Tnmpd)$. Then the total time complexity of BoostNyström is $O(Tnmpd + pm^2n)$. Since usually $Td \ll n$ and $mp \ll n$, the complexity BoostNyström is much smaller than the $O(n^3)$ time complexity usually encountered in kernel methods.

Since in each iteration, computing the approximation errors of subsets $S_r (r = 1, 2, \cdots, \frac{m(p-t+1)}{d})$ are independent, they can be computed in parallel. Then the time complexity of BoostNyström becomes $O(Tnd^2 + pm^2n)$.

The space requirement is the same as that for doing a standard Nyström approximation on m columns. The complexity is $O(nm + m^2)$, which is linear with data size.

4.3.4 Error Bound

To derive an error bound of BoostNyström, we consider a simplified model. Assume that among the mp columns of S, there exist $\hat{p}$ ($1 \leq \hat{p} < p$) and $\hat{\varepsilon}$ ($\hat{\varepsilon} > 0$), such that $\| \mathbf{K} - \tilde{\mathbf{K}}_{r_1} \| < \hat{\varepsilon} < \| \mathbf{K} - \tilde{\mathbf{K}}_{r_2} \|, (\forall r_1 \text{ and } r_2, 1 \leq r_1 \leq \hat{p}, \hat{p}+1 \leq r_2 \leq p)$. $\hat{\varepsilon}$ is the approximation error $\| \mathbf{K} - \tilde{\mathbf{K}}_{ot} \|_2$, and $\tilde{\mathbf{K}}_{ot}$ is the standard Nyström approximation given m sampled columns S_{ot} other than the mp columns in S. Given the adaptive step in BoostNyström, which selects the best subsets of the sampled columns at each iteration, with high probability, there exist sampled columns S_{ot} that satisfy the above condition.

In the proof of the error bound for BoostNyström method, we need the generalization of McDiarmid's concentration bound to sampling without replacement [29] [10], and the error bounds of Ensemble Nyström method first introduced in [27]. The error bounds of Ensemble Nyström are denoted as $\mathcal{B}_2^{ens}$ for the norm-2 error bound, and $\mathcal{B}_F^{ens}$, the Frobenius error bound. We denote K_{max} the maximum diagonal entry of $\mathbf{K}$, $K_{max} = \max_i K_{ii}$, and $d_{max}^{\mathbf{K}}$ the distance $\max_{ij} \sqrt{K_{ii} + K_{jj} - 2K_{ij}}$. We define the selection matrix corresponding to a sample of mp columns as the matrix $\mathbf{L} \in \mathcal{R}^{n \times mp}$. Then $L_{ij} = 1$ if the ith column of $\mathbf{K}$ is among those sampled, and $L_{ij} = 0$ otherwise.

Theorem Let S be a sample of pm columns decomposed into p subsets of size m, $S_1, \cdots, S_p$ at iteration T. For $r \in [1, p]$, let $\tilde{\mathbf{K}}_r$ denote the rank-k Nyström approximation of $\mathbf{K}$ based on the sample S_r, and let $\mathbf{K}_{(k)}$ denote the best rank-k approximation of $\mathbf{K}$. Assume there exist $\hat{p}, (1 \leq \hat{p} < p)$ and $\hat{\varepsilon}, (\hat{\varepsilon} > 0)$, such that $\| \mathbf{K} - \tilde{\mathbf{K}}_{r_1} \| < \hat{\varepsilon} < \| \mathbf{K} - \tilde{\mathbf{K}}_{r_2} \|$, $1 \leq r_1 \leq \hat{p}, \hat{p} + 1 \leq r_2 \leq p$. $\hat{\varepsilon}$ is the approximation error $\| \mathbf{K} - \tilde{\mathbf{K}}_{ot} \|_2$, and $\tilde{\mathbf{K}}_{ot}$ is a standard Nyström approximation given S_{ot}, m sampled columns other than the mp columns in S. Then, with probability at least $1 - \delta$, the following inequalities hold for any set of samples S of size mp and S_{ot} of size m that satisfy the above assumption and for any μ in the simplex Δ ($\Delta = \{\mu \in \mathcal{R}_p \colon \mu \geq 0 \bigwedge \sum_{r=1}^{p} \mu_r = 1\}$) and $\tilde{\mathbf{K}}^{boost} = \sum_{r=1}^{p} \mu_r \tilde{\mathbf{K}}_r$:

$$\| \mathbf{K} - \tilde{\mathbf{K}}^{boost} \|_2 < \mathcal{B}_2^{ens} \tag{4.14}$$

$$\| \mathbf{K} - \tilde{\mathbf{K}}^{boost} \|_F < \mathcal{B}_F^{ens} \tag{4.15}$$

Proof: We replace subset S_1 with S_{ot}, the subset of m sampled columns other than $S_1, S_2, \cdots, S_p$. By definition of $\tilde{\mathbf{K}}^{boost}$, the following holds:

$$
\begin{aligned}
\| \mathbf{K} - \tilde{\mathbf{K}}^{boost} \|_2 &= \| \sum_{r=1}^{p} \mu_r (\mathbf{K} - \tilde{\mathbf{K}}_r) \|_2 \\
&\leq \sum_{r=1}^{p} \mu_r \| \mathbf{K} - \tilde{\mathbf{K}}_r \|_2 \\
&< \mu_1 \| \mathbf{K} - \tilde{\mathbf{K}}_{ot} \|_2 \\
&\quad + \sum_{r=2}^{p} \mu_r \| \mathbf{K} - \tilde{\mathbf{K}}_r \|_2 \\
&\leq \mu_1 (\| \mathbf{K} - \mathbf{K}_{(k)} \|_2 \\
&\quad + 2 \| \mathbf{X}\mathbf{X}^T - \mathbf{Z}_{ot}\mathbf{Z}_{ot}^T \|_2) \\
&\quad + \sum_{r=2}^{p} \mu_r (\| \mathbf{K} - \mathbf{K}_{(k)} \|_2 \\
&\quad + 2 \| \mathbf{X}\mathbf{X}^T - \mathbf{Z}_r\mathbf{Z}_r^T \|_2) \\
&= \| \mathbf{K} - \mathbf{K}_{(k)} \|_2 + \\
&\quad 2 \| \mathbf{X}\mathbf{X}^T - \mathbf{Z}_{ot}\mathbf{Z}_{ot}^T \|_2 + \\
&\quad \sum_{r=2}^{p} 2 \| \mathbf{X}\mathbf{X}^T - \mathbf{Z}_r\mathbf{Z}_r^T \|_2
\end{aligned}
\tag{4.16}
$$

where $\mathbf{Z}_r = \sqrt{\frac{n}{m}}\mathbf{X}\mathbf{L}_r$, and $\mathbf{L}_r$ is the selection matrix for S_r. $\mathbf{Z}_{ot} = \sqrt{\frac{n}{m}}\mathbf{X}\mathbf{L}_{ot}$, and $\mathbf{L}_{ot}$ is the selection matrix for S_{ot}. Let $\bar{S}$ denote the sampled set $S_{ot}, S_2, \cdots, S_p$, and let S' be a sampled set with one column different with $\bar{S}$. Let $\phi(\bar{S}) = \| \mathbf{X}\mathbf{X}^T - \mathbf{Z}_{ot}\mathbf{Z}_{ot}^T \|_2 + \sum_{r=2}^{p} 2 \| \mathbf{X}\mathbf{X}^T - \mathbf{Z}_r\mathbf{Z}_r^T \|_2$. Then the following inequality holds:

$$
| \phi(S') - \phi(\bar{S}) | \leq \frac{2n}{m} \mu_{max} d_{max}^{\mathbf{K}} K_{max}^{\frac{1}{2}} \tag{4.17}
$$

where $\mu_{max} = \max_{i=1}^{p} \mu_i$. The expectation of $\phi(\bar{S})$ is bounded by:

$$E[\phi(\bar{S})] \leq \frac{n}{\sqrt{m}} K_{max} \tag{4.18}$$

(4.17) and (4.18) are derived in the same way as in [27]. Plugging (4.17) and (4.18) into the generalization of McDiarmid's concentration bound, we derive the upper bound for (4.16) as $\mathcal{B}_2^{ens}$. $\| \mathbf{K} - \tilde{\mathbf{K}}^{boost} \|_2$ is strictly less than $\mathcal{B}_2^{ens}$. In the same fashion, we obtain the Frobenius error bound $\| \mathbf{K} - \tilde{\mathbf{K}}^{boost} \|_F$, which is strictly less than $\mathcal{B}_F^{ens}$.

[27] shows that the error bound of Ensemble Nyström method is lower than the normal Nyström approximation. We show that the error bound of BoostNyström is strictly lower than that of Ensemble Nyström method. Thus, BoostNyström is a better kernel matrix approximation tool as compared to state-of-art algorithms. The superior performance is demonstrated experimentally in the following section.

4.4 Experiments

In this section, we present experimental results that illustrate the performance of the BoostNyström method. We work with data sets listed in Table 4.1.

Throughout the experiments, we measure the accuracy of a low-rank approximation $\tilde{\mathbf{K}}$ by calculating the relative error in the following quantity:

$$\%error = \frac{\| \mathbf{K} - \tilde{\mathbf{K}} \|_F}{\| \mathbf{K} \|_F} \times 100 \tag{4.19}$$

For each data set, we set the number of sampled columns in standard Nyström approximation to $m = 0.03n$, where n is the total number of data points. d is shown in Figure 4.3. Rank k is set to be the rank of $\mathbf{W}$ in (4.1). The number of approximations, p, was varied from 2 to 20, and $T = p$. We sampled an additional

Table 4.1: Properties of data sets, n and dim are the number of data points and features respectively.

Dataset	Type of data	n	dim
MNIST-10K [a]	Hand written digits	10000	784
S. Ceravisiae [b]	proteins	4728	16
Abalone [c]	Physical measurements	4177	8
Dexter [c]	Documents	2000	20000
Isolet [c]	Voice	6238	617
USPS [d]	Hand written digits	7291	256

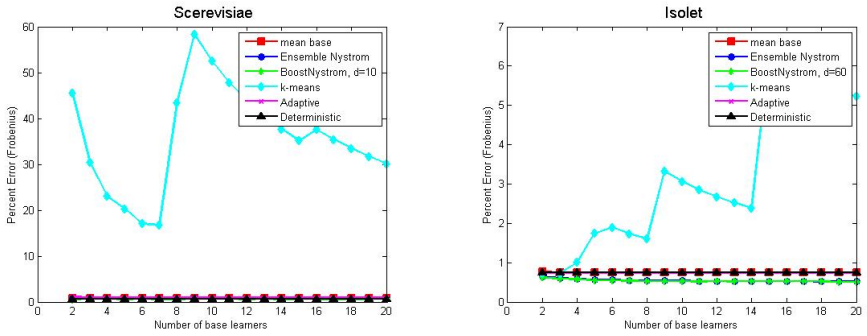
[a] http://yann.lecun.com/exdb/mnist
[b] S.Ceravisiae is from [20]
[c] http://archive.ics.uci.edu/ml/index.html
[d] USPS is from [21]

set of m columns as validation set V, on which the approximation error of each subset is derived, and V is fixed with varying values of p.

We run BoostNyström on each dataset with the sampled columns in Ensemble Nyström approximation as initial samples. The average error (Frobenius error) of BoostNyström is shown in Figure 4.3. It is compared to the following well-known Nyström methods: (a) Standard Nyström approximation with uniform sampling of m columns without replacement. (b) Ensemble Nyström approximation with uniform sampling of mp columns without replacement. (c) K-means based Nyström method [57]. (d) The adaptive sampling method in [28]. (e) The deterministic method in [3]. The average errors of p approximations are reported for (a), (c) and (d).

Figure 4.3 shows that in all data sets, BoostNyström with large p outperforms all other algorithms except in Abalone data. In Abalone data, the K-means based and the adaptive methods are better than BoostNyström. However, the K-means based method is unstable, with good error rate in Abalone, but much worse than

BoostNyström in Isolet and S. Cerevisiae. The adaptive algorithm is unstable too, with best error rate in Abalone, but poor performance in S. Cerevisiae and USPS. We did not draw the error curve of K-means based method in Figure 4.3(b) and Figure 4.3(e), nor the error curve of the deterministic method in Figure 4.3(f), because they are so large that they make other curves undistinguishable. The error rate of the deterministic method in USPS is 9.93%. We draw all curves for S. Cerevisiae and Isolet data sets in Figure 4.2.

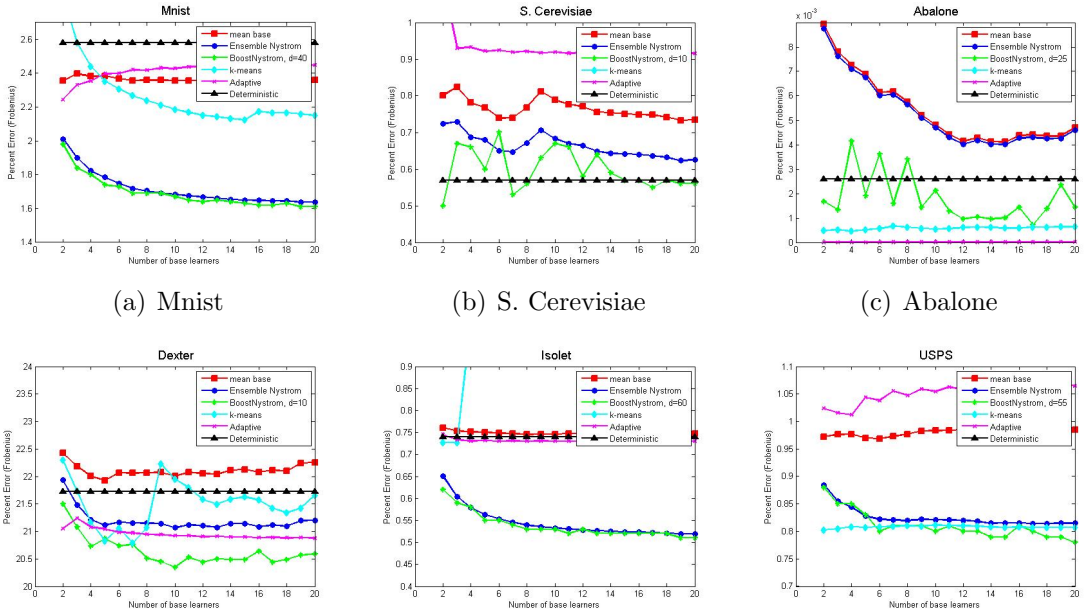


(a) Scerevisiae (b) Isolet

Figure 4.2: Percent error of Scerevisae, Isolet and USPS

4.5 Conclusions and Future Work

We propose a new perspective on the Nyström approximation. The new perspective relates the quality of the Nyström approximation with the subspace spanned by the sample columns. Based on the perspective, we propose BoostNyström, a novel adaptive Nyström approximation method. The BoostNyström algorithm is justified by our novel perspective and has an error bound guaranteed to be lower than that of Ensemble Nyström method. The experimental results show the effectiveness of



(a) Mnist (b) S. Cerevisiae (c) Abalone

(d) Dexter (e) Isolet (f) USPS

Figure 4.3: Percent error in Frobenius norm for base Nyström method, Ensemble Nyström method, BoostNyström, K-means based Nyström approximation, the adaptive Nyström approximation and the deterministic method

BoostNyström. In the future, BoostNyström can be applied to graph-based learning algorithms, such as Graph-OCL.

Chapter 5

General Conclusions

This dissertation has focused on three areas: (1) Supervised learning of high dimensional data. (2) Graph-based one class learning. (3) Nyström method for large size data. Each of the three areas has extensive practical applications. We have developed several state-of-art algorithms to solve each problem.

For supervised learning of high dimensional data, a novel algorithm BDLDA is improved and applied to gene expression data. BDLDA achieves a balance of bias-variance tradeoff, and has provided superior performance compared with other classifiers. Treelets is then used to transform the data before BDLDA, as a step to concentrate more energy near the diagonal positions of the covariance matrix, and thus improve the performance of BDLDA.

For one class learning problem, we have used graph-based ranking method as a first step (identifying outliers) in the algorithm. As far as we know, this is the first time graph-based method has been used in one class learning. One important advantage of our proposed algorithm is its selection of the constant parameter. This frees us of time consuming model selection procedure, like cross validation. This dissertation has provided theoretical proof supporting the parameter selection.

The novel adaptive Nyström method is mainly proposed to solve the scalability issue of the graph-based algorithms. We have proposed a new perspective on the Nyström approximation. The new perspective relates the quality of the Nyström approximation with the subspace spanned by the sampled columns. Based on the perspective, we propose BoostNyström, a novel adaptive Nyström approximation method. The BoostNyström algorithm is justified by our novel perspective. The experimental results show the effectiveness of BoostNyström.

In the future, efforts could be taken to reduce the complexity of BoostNyström to make it more practical in real use. BoostNyström method could be applied to the graph-based one class learning algorithm to make it scalable to large sample size.

Chapter 6

Publications

Lingyan Sheng, Antonio Ortega, Roger Pique-Regi, Shahab Asgharzadeh, "Treelets as feature transformation tool for block diagonal linear discrimination" in *Proceedings of IEEE International Workshop on Genomic Signal Processing and Statistics 2009*

Lingyan Sheng and Antonio Ortega, "Graph-based partially supervised learning of documents" in *Proceedings of IEEE International Workshop on MACHINE LEARNING FOR SIGNAL PROCESSING 2011*

Lingyan Sheng and Antonio Ortega, "Pixel prediction by context based regression" in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing 2012*

Lingyan Sheng and Antonio Ortega, "A Novel Adaptive Nyström Approximation" in *Proceedings of IEEE International Workshop on MACHINE LEARNING FOR SIGNAL PROCESSING 2012*

Lingyan Sheng and Antonio Ortega, "Analysis of Graph-based Ranking and Application to One Class Learning", under preparation of *The IEEE Transactions on Pattern Analysis and Machine Intelligence*

References

[1] Dimitris Achlioptas, Frank McSherry, and Bernhard Schlkopf. Sampling techniques for kernel methods. In *Annual advances in neural information processing systems 14: Proceedings of the 2001 conference*, pages 335–342. MIT Press, 2001.

[2] Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC '01, pages 619–626, New York, NY, USA, 2001. ACM.

[3] Mohamed-Ali Belabbas and Patrick J. Wolfe. Spectral methods in machine learning and new strategies for very large datasets. In *Proceedings of the National Academy of Sciences of the USA*, volume 106(2), pages 369–374, January 2009.

[4] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *COLT*, volume 3120, pages 624–638, 2004.

[5] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006.

[6] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 217–235, London, UK, UK, 1999. Springer-Verlag.

[7] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 19–26, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[8] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.

[9] Anthony Brew, Marco Grimaldi, and Pádraig Cunningham. An evaluation of one-class classification techniques for speaker verification. *Artificial Intelligence Review*, 27:295–307, 2007.

[10] Corinna Cortes, Mehryar Mohri, Dmitry Pechyony, and Ashish Rastogi. Stability of transductive regression algorithms. *Proceedings of the 25th International Conference on Machine Learning*, pages 176–183, 2008.

[11] Corinna Cortes and Vladimir Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[12] Ramon Diaz-Uriarte and Sara Alvarez de Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3, 2006.

[13] S. Dudoit, J. Fridlyand, and T.P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457):77–87, 2002.

[14] Dinesh Singh et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, I(2):203–209, Mar 2002.

[15] U. Alon et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci U S A*, 96(12):6745–6750, Jun 1999.

[16] Ahmed Farahat, Ali Ghodsi, and Mohamed Kamel. A novel greedy algorithm for nyström approximation. *Journal of Machine Learning Research*, 15:269–277, 2011.

[17] Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2:243–264, 2002.

[18] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[19] Y Guo, T Hastie, and R Tibshirani. Regularized linear discriminant analysis and its application in microarrays. *Biostatistics*, pages 86–100, 1 2007.

[20] Adam M Gustafson, Evan S Snitkin, Stephen Cj Parker, Charles DeLisi, and Simon Kasif. Towards the identification of essential genes using targeted genome sequencing and comparative analysis. *BMC Genomics*, 7:265, 2006.

[21] T Hastie, R Tibshirani, and JH Friedman. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. Springer, 2001.

[22] Thorsten Joachims. *Making large-scale support vector machine learning practical*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[23] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[24] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *In ICML*, pages 290–297, 2003.

[25] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[26] Mark W. Koch, Mary M. Moya, Larry D. Hostetler, and R. Joseph Fogler. Cueing, feature discovery, and one-class learning for synthetic aperture radar automatic target recognition. *Neural Netw.*, 8(7-8):1081–1102, 1995.

[27] Sanjiv Kumar, M Mohri, and A Talwalkar. Ensemble nyström method. *Advances in Neural Information Processing Systems*, 5:1–9, 2009.

[28] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. On sampling-based approximate spectral decomposition. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 553–560, 2009.

[29] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling techniques for the nyström method. *Journal of Machine Learning Research - Proceedings Track*, 5:304–311, 2009.

[30] Ken Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ICML95)*, 1995.

[31] Ann B. Lee, Boaz Nadler, and Larry Wasserman. Treelets - an adaptive multiscale basis for sparse unordered data. *Annals of Applied Statistics*, 2(2), 2008.

[32] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 587–592, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[33] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. Partially supervised classification of text documents. In *ICML02*, pages 387–394, 2002.

[34] Yvette Mallet, Danny Coomans, Jerry Kautsky, and Olivier De Vel. Classification using adaptive wavelets for feature extraction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:1058–1066, 1997.

[35] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Test Categorization*, pages 41–48. AAAI Press, 1998.

[36] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[37] Tao Peng, Wanli Zuo, and Fengling He. Svm based adaptive learning method for text classification from positive and unlabeled documents. *Knowl. Inf. Syst.*, 16:281–301, 2008.

[38] R Pique-Regi and A Ortega. Block diagonal linear discriminant analysis with sequential embedded feature selection. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2006*, volume 5, pages V–V, 2006.

[39] R. Pique-Regi, A. Ortega, and S. Asgharzadeh. Sequential diagonal linear discriminant analysis (seqDLDA) for microarray classification and gene identification. In *CSBW*, pages 112–116, 2005.

[40] John C. Platt. Fast embedding of sparse music similarity graphs. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

[41] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[42] DG Stork RO Duda, PE Hart. *Pattern Classification*. Wiley-Interscience, 2 edition, 2000.

[43] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10:1299–1319, 1998.

[44] W. M. Shaw, Jr. On the foundation of evaluation. *Journal of the American Society for Information Science*, 37(5):346–348, 1986.

[45] Lingyan Sheng and Antonio Ortega. Graph based partially supervised learning of documents. In *IEEE International Workshop on Machine Learning for Signal Processing*, 2011.

[46] Lingyan Sheng and Antonio Ortega. A novel adaptive nyström approximation. In *IEEE International Workshop on Machine Learning for Signal Processing*, 2012.

[47] Lingyan Sheng, Antonio Ortega, Roger Pique-Regi, and Shahab Asgharzadeh. Treelets as feature transformation tool for block diagonal linear discrimination.

In *IEEE International Workshop on Genomic Signal Processing and Statistics*, 2009.

[48] Lingyan Sheng, Roger Pique-Regi, Shahab Asgharzadeh, and Antonio Ortega. Microarray classification using block diagonal linear discriminant analysis with embedded feature selection. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009.

[49] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 824–831, New York, NY, USA, 2005. ACM.

[50] Alex J. Smola and Bernhard Schökopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 911–918, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[51] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952. MIT Press, 2002.

[52] R Tibshirani, T Hastie, B Narasimhan, and G Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc Natl Acad Sci U S A*, 99(10):6567–6572, May 2002.

[53] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2 edition, 2000.

[54] Wei Wang and Jiong Yang. Mining high-dimensional data. In *Data Mining and Knowledge Discovery Handbook*, pages 793–799. Springer US, 2005.

[55] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[56] Hwanjo Yu, Jiawei Han, and Kevin Chen chuan Chang. Pebl: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16:70–81, 2004.

[57] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239, 2008.

[58] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press, 2004.

[59] Dengyong Zhou and Bernhard Schölkopf. Learning from labeled and unlabeled data using random walks. In *Pattern Recognition, Proceedings of the 26th DAGM Symposium*, pages 237–244. Springer, 2004.

[60] Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 16*. MIT Press, 2003.

[61] Xueyuan Zhou, Mikhail Belkin, and Nathan Srebro. An iterated graph laplacian approach for ranking on manifolds. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 877–885, New York, NY, USA, 2011. ACM.

[62] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[63] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.