This is to certify that the
dissertation entitled

Label Propagation for Classification and Ranking

presented by

Ming Wu

has been accepted towards fulfillment
of the requirements for the

Doctoral degree in Computer Science

Major Professor's Signature

July 16, 2007

Date

LABEL PROPAGATION FOR CLASSIFICATION AND RANKING

By

Ming Wu

A DISSERTATION

Submitted to
Michigan State University
In partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

2007

ABSTRACT

LABEL PROPAGATION FOR CLASSIFICATION AND RANKING

By

Ming Wu

Label Propagation has been proven to be an effective semi-supervised learning approach in many applications. The key idea behind label propagation is to first construct a graph in which each node represents a data point and each edge is assigned a weight often computed as the similarity between data points, then propagate the class labels of labeled data to neighbors in the constructed graph in order to make predictions. This dissertation is a comprehensive study of the label propagation approaches in different directions including *Relation Propagation*, *Rank Propagation* and *Propagation over Directed Graphs*.

Most previous works on label propagation propagate information among a *single* type of objects. However, many applications involve multiple types of objects. Inspired by the assumption that the correlation among different types of objects can be very helpful in many cases, a generalized framework for *Relation Propagation* is proposed to explore the correlation in semi-supervised learning. The key idea behind relation propagation is to construct a graph which involves multiple types of objects and then propagate the relation among different types of objects in this graph. The framework for *Relation Propagation* is applied to multi-label learning (classification problems) and collaborative filtering (ranking problems). Empirical results show that relation propagation is a more effective approach in comparison with the previous approaches in label propagation.

It is very important to study the label propagation approaches for ranking problems due to the existing challenges for label propagation. First, it may not be appropriate to propagate class labels (ordinal values) as numerical values in classification problems. It seems more reasonable to cast the problem into a ranking problem in which the class labels are

converted to pairwise preferences between classes for each example and then the preferences are propagated. Second, most previous studies require absolute labels for learning which are often hard to obtain. Instead, relative ordering information is more easily available. Traditional label propagation approaches may not fit with ranked data. Inspired by these challenges, a *Rank Propagation* framework is proposed for supervised learning. The key idea behind *Rank Propagation* is to propagate the given preference judgements, instead of the true labels, from the labeled data to unlabeled data and compute the preference matrices for unlabeled data whose principal eigenvectors correspond to the class assignments. The application of this framework is presented in a multi-label categorization task with multiple datasets. The empirical results show that *Rank Propagation* is an effective approach in comparison with other commonly used supervised learning approaches.

Most studies in label propagation focus on using the undirected graphs. Motivated by the assumption that directed graph may better capture the nature of data, a framework for *Propagation over Directed Graphs* is proposed for utilizing the directed graph in propagation. The question involved in this approach is how to construct and utilize a directed graph. Two asymmetric weight measures, namely KL divergence-based measure and asymmetric cosine similarity, are proposed in order to construct a directed graph. To utilize the directed graphs, one common method is to convert the directed graphs into undirected ones and then apply a standard label propagation approach to the converted undirected graphs. A random walk related method is discussed for this conversion. The application of this framework is presented in a binary classification task and a multi-label classification task. The empirical results show the effectiveness of *Propagation over Directed Graphs* in comparison with other approaches.

In summary, this dissertation discusses the proposed approaches in label propagation, namely *Relation Propagation*, *Rank Propagation* and *Propagation over Directed Graphs*, in order to address the existing challenges in this area. Empirical studies show that the proposed approaches achieve promising performance in given scenarios.

ACKNOWLEDGMENTS

This thesis would not have been possible without the support of many people. Many thanks to my advisor, Rong Jin, who read my numerous revisions and helped make some sense of the confusion. Also thanks to my committee members, Eric Torng, Pang-Ning Tan, and Sarat Dass, who offered guidance and support. Special thanks to my fiancé, Aron White, who put up with me and provided so much love and support. Finally, thanks to my parents, and many friends who walked with me every step on this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

As a graph-based approach, *Label Propagation* belongs to the family of semi-supervised learning and has been proved to be effective for both text categorization and information retrieval [47, 10, 57, 50]. Assuming the objects in question are documents, the key idea of *Label Propagation* is as follows: first, view each document as a node in a connected graph and each edge of the graph is associated with a weight which is proportional to the similarity between the two connected nodes/documents; then, the confidence scores for all the classes of the unlabeled documents are estimated by propagating the class labels of the labeled documents through the weighted edges; finally, the estimated confidence scores are either used to determine the appropriate categories for unlabeled documents in text categorization, or used to rank the documents in information retrieval. One of the key components in label propagation is the document similarity. It is usually calculated based on the content of documents, using either the dot product between two document-term vectors or the RBF kernel function. It can also be computed using the side information, such as the hyperlinks among web pages [17].

A number of approaches have been developed in the past for label propagation. including the approach based on the Harmonic function [57], Gaussian random field [47], and the Green function approach [50]. In the meantime, a number of propagation approaches have

been developed in information retrieval, including the pagerank algorithm [8], the HITS algorithm [27], and retrieval based on implicit link [48, 29].

This dissertation explores and extends the idea of label propagation in several different avenues, namely *Relation Propagation*, *Rank Propagation* and *Propagation over Directed Graphs*.

Relation Propagation. First, the generalized framework for *Relation Propagation* will be exploited and evaluated. Most previous works on label propagation propagate information among a *single* type of objects [57, 51, 23]. However, in the real world, many applications involve multiple types of objects. For instance, document categorization tasks involve two types of objects: documents and categories. In these scenarios, the information needs to be propagated among the objects of different types in order to utilize the correlation among the objects of different types which can provide useful information for classification and ranking problems. Inspired by the assumption that correlation among the different types of objects can be very helpful in many cases, a generalized framework is proposed for *Relation Propagation*. The difference between our work and previous research is that the *Relation Propagation* framework allows the relationship between multiple types of objects to be propagated and thus improve the performance.

To better understand this framework, consider the scenario which involves two types of objects: A and B. The key idea behind *Relation Propagation* is to first construct a graph in which each node is a object pair consisting of one object from each type and then propagate relationship between A and B in this graph. This framework of *Relation Propagation* can be easily applied to many applications. For instance, in a document categorization task, two types of objects are documents and categories. The graph can be constructed as: each node is a document-category pair whose label is the membership of the document belonging to the category; each edge is associated with a weight which can be defined using some similarity measure. Then, the membership information can be propagated through this

graph. The resulting scores for each document with respect to all categories will lead to a ranking list of all categories for this document. The framework can also be easily extended to the scenario with more than two types of objects.

In order to show the effectiveness of the relation propagation model, it is applied to two kinds of tasks: text categorization tasks and collaborative filtering tasks. The text categorization task is studied with an image retrieval dataset in which two types of objects are images and categories. The membership of an image belonging to a category is the information to be propagated. The collaborative filtering problem is studied with movie ratings and book recommendation datasets in which users and items are two types of objects and the ratings information is propagated. Empirical results show that the *Relation Propagation* achieves very promising performance in many applications in comparison with other state-of-art techniques.

Rank Propagation. The ranking problem has always been an important subject in many areas such as machine learning and statistics. A ranking problem often requires the ranked data as the input with the goal of generating a ranking function. Many previous works have been done in this area [2, 22, 43, 12, 32, 31]. Although label propagation, as an effective semi-supervised learning technique, has shown promising performance in many applications, it is quite natural to explore the label propagation approaches on ranked data because of the existing challenges in label propagation.

One of the challenges is related to propagating directly the class labels. As described above, the label propagation approaches address all the problems by first propagating the class labels and then generating ranking lists based on the resulting propagation scores computed by the algorithm. This may not be an appropriate approach in certain cases (e.g., classification) since there is no ordering information among class labels. For example, consider the numerical class labels $1, 2, 3, \cdots$. In most cases, the numbers just provide the labeling for classes instead of the ordering information. Class 1 is not greater or smaller

than class 2. When the class labels are propagated directly, the ordering information will be incorrectly introduced into the propagation process. One way to avoid this problem is to first convert the class labels into pairwise preferences between classes for each training example and then propagate these preferences. Clearly this idea can be cast to a ranking problem.

Another challenge is the difficulty of obtaining the exact labels of the labeled data. Most previous studies require the *absolute* label values or numeric values to be given in order to learn the ranking function, which is often not practical. The absolute labeling information is usually hard to obtain or is very costly and time consuming. In contrast, the partial *relative* ordering information is more easily available for training in some cases. For instance, in a movie ratings dataset, the users provide the ratings which indicate the preferences of certain movies over other movies. Traditional label propagation may not fit in these applications because they are required to propagate the absolute labels.

To address these problem, a *Rank Propagation* framework is proposed for multi-label learning. The discussion starts with the definition of a general *ranking* problem whose input is the preference judgements, followed by the presentation of the *Rank Propagation* framework of supervised learning. The key idea behind *Rank Propagation* is to propagate the preference judgements (relative ordering) of the labeled data between the labeled data and unlabeled data, instead of propagating the true labels of the labeled data. The preference judgements can be directly given by the datasets or computed from the labeles of the given data. Consider the example of text categorization. For each document, a preference matrix can be computed with respect to the categories and each entry in the preference matrix is the preference between two categories of this document. The preference judgements from the labeled data are propagated between the labeled documents and the unlabeled documents. The goal of *Rank Propagation* is to achieve the preference matrices for unlabeled data whose principal eigenvectors are proved to correspond to the class assignments.

The application of this framework is presented in a multi-label categorization task with

movie categorization, image retrieval and gene categorization datasets. The empirical results show that *Rank Propagation* is an effective approach in comparison with other commonly used supervised learning approaches, and thus verify our assumption that propagating the relative ordering information can be more helpful for solving the problems.

Propagation over Directed Graphs. As mentioned above, label propagation often involves a graph constructed from the given datasets. Most previous studies in label propagation focus on using the undirected graphs [57, 51]. However, directed graphs can be more appropriate in describing the given datasets in many cases. For instance, in the web page categorization scenario, it's often better to view the hyperlinks as directed edges instead of undirected edges in the graph. A number of works have been devoted to the graph-based approaches on the directed graph [50, 52]. Motivated by the assumption that directed graph may better capture the nature of the data, a framework for *Propagation over Directed Graphs* is proposed. The general idea of this approach is to first construct a directed graph from a given dataset, then convert this directed graph into an undirected graph and finally propagate the labeling information over this converted graph using some standard propagation scheme.

There are two questions involved in this approach. The first question is how to construct a directed graph from the given data. In label propagation, a matrix is often computed in which each entry is the pairwise relationship between two data points. Based on this matrix, the graph is constructed in which each node represents a data point and each edge is associated with a weight equal to the corresponding pairwise relationship in the matrix. Evidently symmetric weight measures result in the undirected graphs since two edges connecting two nodes carry the same weights while asymmetric weight measures lead to the directed graphs. Based on this observation, two asymmetric weight measures are proposed, namely KL divergence-based measure and asymmetric cosine similarity, in order to construct a directed graph. The second question is how to utilize the directed graphs. One

straightforward way is to convert a directed graph into an undirected one and then apply the existing label propagation approaches to the converted undirected graph. The regularization framework in [52] is discussed for this conversion. Spectral Graph Transducer (SGT) is used as the standard propagation scheme in the proposed approach because of its proven effectiveness in various applications.

The application of this framework is presented in a binary classification task and a multi-label classification task. In the binary classification task, the approach of *Propagation over Directed Graphs* is compared with propagation using undirected graphs and other state-of-art classification techniques with multiple datasets. The empirical results show the effectiveness of *Propagation over Directed Graphs* in comparison with other approaches. The analysis of why directed graphs outperform other approaches verifies the assumption that directed graphs can better express the nature of the data in some cases. A brief empirical study is given for the multi-label classification in which the proposed approach is compared with only the label propagation approach based on harmonic functions and Gaussian fields. The goal is to show that using directed graphs is more effective than using undirected graphs from a different point of view.

It is important to note that this dissertation focuses on the rank-related evaluation metrics for all the methods, which is different from the commonly used metrics in traditional classification and regression approaches. Traditionally, researchers often evaluate the effectiveness of the label propagation approaches by classification accuracy. This requires an algorithm to generate the exact labels or numerical values in order to compare with the true assignments of classes. Since most label propagation approaches generate the scores which can only provide the ranking information for the unlabeled data, it is necessary to convert these scores into true labels. However, by converting from the ranking into the binary classification decision, it always involves the selection of threshold which is often determined empirically and prone to inaccuracy. In order to evaluate the essential effectiveness of the

label propagation approaches without the complication of the threshold, this dissertation focuses on the label propagation approaches for ranking problems. Later chapters will discuss how label propagation can be applied to many scenarios by solving a ranking problem with the presentation of the proposed works.

The rest of this dissertation is organized as follows: Chapter 2 introduces the general idea of label propagation with the existing related works and discusses in detail two label propagation approaches which are closely related to our work. Chapter 3 presents the *Relation propagation* approach which includes both the general framework of this approach and its applications for the multi-label learning task and the collaborative filtering task with the empirical results. Several drawbacks of the label propagation approaches for ranking problems will be discussed at the end of the chapter. Chapter 4 defines the general ranking problem and proposes the *Rank Propagation* framework for supervised learning which propagates the relative ordering information instead of true labels of the labeled data. Chapter 5 proposes the framework of *Propagation over Directed Graphs* and discusses its applications of binary classification and multi-label classification along with an analysis of the reasons why directed graphs can better express the nature of the data. Chapter 6 concludes this dissertation with summarization and future work.

Chapter 2

Label Propagation : A Graph-based Approach

In many traditional approaches to machine learning, the learner is trained by labeled examples. Labeled examples are often, however, time consuming and expensive to obtain because they require very skilled human annotators. Therefore, the problem of combining labeled and unlabeled examples together in the learning process becomes very important and necessary. The semi-supervised learning utilizes both labeled and unlabeled examples and has attracted an increasing amount of interest. A number of semi-supervised learning approaches have been proposed [41, 4, 23, 5, 51]. Among these approaches, there is one family of techniques which exploit the *manifold structure* of the data that is usually inferred from the pairwise similarity of unlabeled and labeled data points. This family of techniques is referred to as **label propagation** since they propagate the label information from labeled examples to unlabeled examples by using the pairwise similarity. The unlabeled examples are classified by using the labeling information propagated to the unlabeled examples from the labeled examples. The general process of label propagation is to propagate the labels of the labeled data over the graph in which each node represents a data point and each edge connecting two nodes is associated with a user defined weight. Figure 2.1 shows an

example of the graph for label propagation.

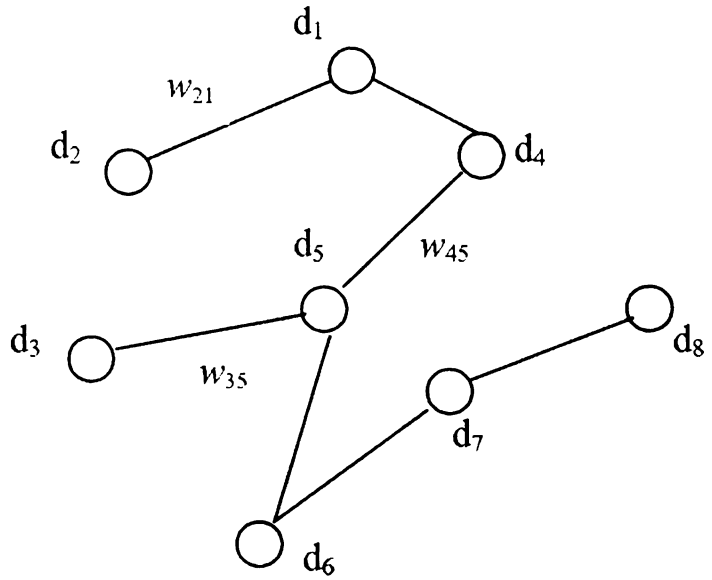


Figure 2.1: An Example of the Connected Graph for Label Propagation

2.1 Related Works

Nearly all label propagation approaches are based on the prior assumption of consistency, which means: (1) nearby points are likely to have the same label; and (2) points on the same structure (typically referred to as a cluster or a manifold) are likely to have the same label. The difference among various algorithms for label propagation lies in the way to model the structure of the data and realize the assumption of consistency, and the way to propagate the label information from labeled examples to unlabeled ones.

A number of previous works have been devoted to label propagation [47, 10, 57, 50]. In [57, 54, 56], Gaussian random fields and Harmonic functions-based methods are motivated by the assumption that the label assignments should be smooth over the entire graph. In [51], the local and global consistency method proposes to enforce the smoothness of the class assignments by minimizing the sum of local variations measured at each edge in an undirected graph. In [23], Spectral Graph Transducer, which can be viewed a transduc-

tive version of KNN, propagates the labels of the labeled data by incorporating a quadratic penalty on the labeled data into the minimization problem of normalized graph cuts with constraints. In [4, 5], the authors propose to extend the graph mincuts method for transductive learning. The key idea is to search for a partition of the graph which results in a minimum sum of weights of the cut while agreeing with the labeled data. Local Laplacian embedding methods propose to project the data from the original space to a dimension reduced space and then the classification function can defined on the reduced dimension space [3, 13].

Among various approaches in label propagation, the two most popular ones are the Gaussian fields and Harmonic functions-based approach [57] and the local and global consistency approach [51], which both have been successfully applied to a number of applications. We will present these two approaches in detail due to their resemblance to our proposed framework of *Relation Propagation* in Section 3. The Harmonic function-based approach predicts the class labels by enforcing the smoothness of the class assignments over the entire graph. The smoothness of the class assignments is defined by a quadratic energy function which has a harmonic solution. The local and global consistency approach presents a simple algorithm to achieve a smooth classification function with respect to the intrinsic structure collectively revealed by known labeled and unlabeled points. This algorithm enforces the smoothness by minimizing the sum of local variations measured at each edge in the graph. These two methods are consistent with each other in that (1) both of them achieve the smoothness of label assignment probabilities over the entire graph by minimizing the sum of the variations defined at each edge; (2) the initial labels with the labeled data are retained in some way for both methods. In the following section, the two different label propagation approaches are reviewed in detail.

2.2 Label Propagation based on Gaussian Fields and Harmonic Functions

[57] introduces a label propagation approach based on a random field model defined on a weighted graph over the unlabeled and labeled data, where the weights are given in terms of a similarity function between instances.

This label propagation method adopts *Gaussian* fields over a continuous state space rather than random fields over a discrete label set. This "relaxation" to a continuous rather than discrete sample space results in many attractive properties in that the most probable configuration of the field is unique, characterized in terms of harmonic functions with a closed form solution that can be computed using matrix methods or loopy belief propagation.

2.2.1 Description of the Framework

Assume there are l labeled examples $(x_1, y_1), \cdots, (x_l, y_l)$ and u unlabeled examples $x_{l+1}, \cdots, x_{l+u}$ ($l \ll u$). Let $n = l + u$ be the total number of data examples. Consider the binary classification problem: $y \in \{0, 1\}$. Construct a connected graph $G(V, E)$ where V corresponds to the n data examples with $L = \{1, \cdots, l\}$ corresponding to the labeled examples with labels $\{y_1, \cdots, y_l\}$ and $U = \{l+1, \cdots, l+u\}$ corresponding to the unlabeled examples. The goal is to assign labels to U. Let W denote a $n \times n$ symmetric weight matrix on the edges E of the graph. W can be computed by RBF kernel as:

$$w_{ij} = \exp\left(-\sum_{d=1}^{m} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2}\right) \tag{2.1}$$

where x_{id} is the d-th element for the example x_i represented as a vector $x_i \in \mathbf{R}^m$, and σ_d is length scale hyper parameters for each dimension. Obviously, nearby points in Euclidean space are assigned large edge weight. Other measures are also possible to be adopted if

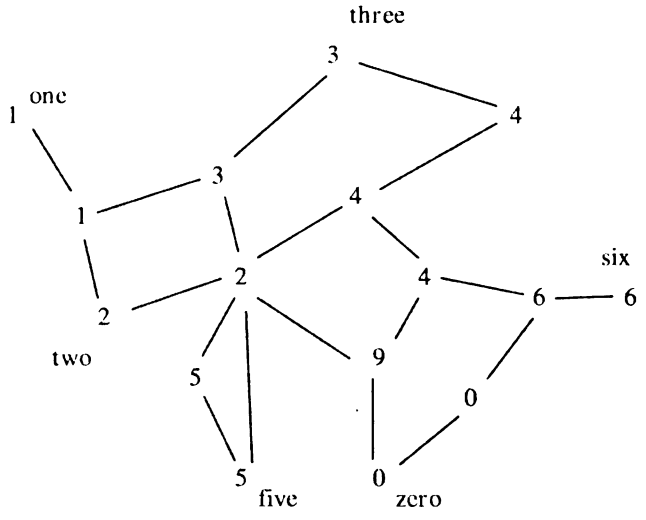


Figure 2.2: The random fields used in this work are constructed on labeled and unlabeled examples. The graph is formed with weighted edges between instances (digits), with labeled data as special "boundary" points and unlabeled points as "interior" points.

they are meaningful for the application as long as they generate non-negative weights.

The goal is to find a real-valued function $f : V \rightarrow \mathbf{R}$ on G and then assign the labels based on f. Also we constrain $f(i) = f_l(i) \equiv y_i$ for all labeled examples. We use f_l to denote the function values for labeled examples and f_u for unlabeled examples. It is intuitive to assume that data points sharing large similarities should have similar label assignments. This assumption leads to the quadratic energy function

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij} \left(f(i) - f(j)\right)^2 \tag{2.2}$$

A probability distribution is assigned on f by forming the Gaussian field $p_\beta(f) = \frac{e^{-\beta E(f)}}{Z_\beta}$, where β is an "inverse temperature" parameter, Z_β is the partition function $Z_\beta = \int_{f|L=f_l} \exp\left(-\beta E(f)\right) df$, which normalizes over all functions constrained to f_l on the labeled data. Clearly minimizing this energy function will lead to a smooth label assignment function f because the minimum value can only be reached by enforcing the similar label probabilities to the pairs of nodes which share large similarities. In other words, each node will finally be assigned a label probability which is consistent with its neighbors and

this leads to a smooth label assignments over the entire graph.

The minimum energy function $f = \text{argmin}_{f|L=f_l} E(f)$ is harmonic. It satisfies $\Delta f = 0$ on unlabeled data points U and is equal to f_l on the labeled examples L. Δ is *combinatorial Laplacian* and the matrix form is $\Delta = D - W$ where $D = \text{diag}(d_i)$ is the diagonal matrix with entries $d_i = \sum_j w_{ij}$ and $W = [w_{ij}]$ is the weight matrix. The harmonic property means that the value of f at each unlabeled data point is the average of f at neighboring points

$$f(j) = \frac{1}{d_j} \sum_{i \sim j} w_{ij} f(i)$$

where $j = l+1, \cdots, l+u$. In other words, $f = Pf$ where $P = D^{-1}W$. According to the maximum principle of harmonic functions [14], a harmonic function f defined on S (S is an open subset of R^n) takes on its maximum value and its minimum value on the boundary. Thus we guarantee $0 < f(j) < 1$ for $j \geq l+1$ since the labeled data are "boundary" points and unlabeled data are "interior" points in the graph.

The harmonic solution can be computed explicitly in matrix form. The weight matrix W can be split into 4 blocks after lth row and column:

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}$$

Letting $f = \begin{bmatrix} f_l \\ f_u \end{bmatrix}$, the harmonic solution $\Delta f = 0$ subject to $f|L = f_l$ can be presented as:

$$f_u = (D_{uu} - W_{uu})^{-1} W_{ul} f_l = (I - P_{uu})^{-1} P_{ul} f_l \tag{2.3}$$

To better understand the above method, we will show its application to document classification task. In order to apply the label propagation method, we relax the discrete class labels to a continuous space. The classification method will be divided into two steps: first,

we estimate the continuous class label scores by propagating the class labels of labeled data over the graph; then based on the continuous class label scores, we can make predictions by an empirically determined threshold.

Let $\mathcal{D} = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n)$ denote the entire collection of documents. Assume that the first n_l documents of $\mathcal{D}$ are labeled by $\mathbf{y}_l = (y_1, y_2, \ldots, y_{n_l})$, and the remaining $n_u = n - n_l$ documents are unlabeled. Each label y_i can either be $+1$ or -1. Let $S = [S_{i,j}]_{n \times n}$ denote the similarity matrix for both labeled and unlabeled documents. To estimate the class labels for the unlabeled document, denoted by $\mathbf{y}_u$, the label propagation approach described above suggested minimizing the following energy function, i.e.,

$$E_s = \sum_{i,j=1}^{n} S_{i,j}(y_i - y_j)^2 = \mathbf{y}^\top L \mathbf{y} \tag{2.4}$$

where $\mathbf{y} = (\mathbf{y}_l^\top, \mathbf{y}_u^\top)^\top$. Matrix $L = [L_{i,j}]_{n \times n}$ is the graph Laplacian for similarity matrix S. It is defined as $L = D - S$ where $D = \text{diag}(D_1, D_2, \ldots, D_n)$ and $D_i = \sum_{j=1}^{n} S_{i,j}$. The optimal solution $\mathbf{y}_u$ for the above problem is approximated in [57] as

$$\mathbf{y}_u = L_{u,u}^{-1} S_{u,l} \mathbf{y}_l. \tag{2.5}$$

where $L_{u,u}$ refers to the part of the graph Laplacian or combinatorial Laplacian L that is only related to the unlabeled documents, and $S_{u,l}$ stands for the similarity matrix between the unlabeled documents and the labeled documents.

2.2.2 Interpretation

The framework can be viewed in several fundamentally different ways. These different viewpoints enrich the understanding and reasoning about this approach to the semi-supervised learning problem.

The harmonic solution can be viewed as a random walk approach. Imagine a particle

walking along the graph G. Starting from an unlabeled data i, the particle moves to a node j with probability P_{ij} after one step. The walk continues until the particle hits a labeled node. Then $f(i)$ is the probability that the particle, starting from node i, hits a labeled node with label 1. This view of the harmonic solution indicates that it is closely related to the random walk approach in [45]. There are two major differences between the harmonic function approach and the random walk approach: first, f is fixed on the labeled examples in the harmonic function approach; second, the harmonic function solution described above is achieved from an equilibrium state, while in [45] the random walk approach depends on the time parameter t.

The solution f can also be viewed from the viewpoint of spectral graph theory. The heat kernel with time parameter t on the graph G is defined as $K_t = e^{-t\Delta}$. Here $K_t(i,)$ is the solution to the heat equation on the graph with initial conditions being a point source at i at time $t = 0$. It was proposed as an appropriate kernel for machine learning with categorical data in [28]. When used in a kernel method such as a support vector machine, the kernel classifier $\hat{f}_t(j) = \sum_{i \in L} \alpha_i y_i K_t(i,j)$ can be viewed as a solution to the heat equation with initial heat sources $\alpha_i y_i$ on the labeled data. The time parameter t must be chosen using an auxiliary technique. The harmonic function approach described in the previous section uses a different approach independent of t, diffusion time. Consider the heat kernel $K_t' = e^{-t\Delta_{uu}}$ on Δ_{uu} where $\Delta_{uu} = D_{uu} - W_{uu}$, the Laplacian restricted to the unlabeled data examples on G. K_t' describes heat diffusion on the unlabeled subgraph with Dirichlet boundary conditions on the labeled nodes. A Dirichlet boundary condition (often referred to as a first-type boundary condition) imposed on an ordinary differential equation or a partial differential equation specifies the values a solution is to take on the boundary of the domain. Clearly Dirichlet boundary conditions fix the values of the function on the labeled data. The Green's function $\mathcal{G}$ can be expressed as:

$$\mathcal{G} = \int_0^\infty K_t' dt = \int_0^\infty e^{-t\Delta_{uu}} dt = (D_{uu} - W_{uu})^{-1} \tag{2.6}$$

The harmonic solution in 2.3 can be written as

$$f_u = \mathcal{G} W_{ul} f_l \tag{2.7}$$

or

$$f(j) = \sum_{i=1}^{l} \sum_{k} y_i w_{ik} \mathcal{G}(k, j) \tag{2.8}$$

The above expression shows that this approach can be viewed as a kernel classifier with the kernel $\mathcal{G}$ and a specific form of kernel machine.

The graph mincut approach proposed in [4] has very interesting and substantial connection to the harmonic function method described here. The starting point for the graph mincut approach is also a weighted graph G, but the learning problem is presented as finding the minimum st-cut, where negative labeled data is connected to a special source node s and positive labeled data is connected to a special sink node t. A minimum st-cut, which is not necessarily unique, minimizes the L^1 objective function $E_1(f) = \frac{1}{2} \sum_{i,j} w_{ij} |f(i) - f(j)|$ and corresponds to a function $f : V \rightarrow \{-1, +1\}$. The solution may not be unique and can be obtained using linear programming. However, its random field model is over the label space $\{-1, +1\}$ and the field is pinned on the labeled entries. This leads to two problems. First, the approximation methods based on rapidly mixing markov chains can not be used; second, multi-label extensions of this approach are generally NP-hard. Instead, the harmonic solution can be computed efficiently using matrix methods, even in the multi-label case and inference for the Gaussian random field can be efficiently and accurately carried out using loopy belief propagation [46].

2.2.3 Learning the Weight Matrix

The weight matrix is an important input in the harmonic function approach described above. Consider the weight matrix given by the equation 2.1, the parameter σ_d is the only parameter determining the weight matrix. By learning σ_d from both labeled and unlabeled data,

the graph structure can be better aligned with the data. One way to learn the parameter σ_d is to minimize the entropy on the unlabeled data.

The average label entropy is used as a heuristic criterion for parameter learning and it is defined as

$$H(f) = \frac{1}{u} \sum_{i=l+1}^{l+u} H_i(f(i)) \tag{2.9}$$

where $H_i(f(i)) = -f(i)\log f(i) - (1 - f(i))\log(1 - f(i))$ is the entropy of the field at the individual unlabeled data point i. The maximum principle of harmonic functions guarantees that $0 < f(i) < 1$ for $i \geq l + 1$. Small entropy implies that $f(i)$ is close to 0 or 1 and this captures the intuition that a good W should result in a confident labeling.

Smoothing can be used to avoid the complication that H has a minimum at 0 as $\sigma_0 \rightarrow 0$. We replace P (by normalizing weight matrix W) with the smoothed matrix $\widetilde{P} = \epsilon\mathcal{U} + (1 - \epsilon)W$, where $\mathcal{U}$ is the uniform matrix with entries $\mathcal{U}_{ij} = 1/(l + u)$.

We use gradient descent to find the hyperparameters σ_d that minimize H. The gradient is computed as

$$\frac{\partial f_u}{\partial \sigma_d} = (I - \widetilde{P}_{uu})^{-1} \left(\frac{\partial \widetilde{P}_{uu}}{\partial \sigma_d} f_u + \frac{\partial \widetilde{P}_{ul}}{\partial \sigma_d} f_l \right)$$

Since the original matrix P is obtained by normalizing W, we have

$$\frac{\partial p_{ij}}{\sigma_d} = \frac{\frac{\partial w_{ij}}{\partial \sigma_d} - p_{ij} \sum_{n=1}^{l+u} \frac{\partial w_{in}}{\partial \sigma_d}}{\sum_{n=1} l + u w_i n}$$

Finally, $\frac{\partial w_{ij}}{\partial \sigma_d} = 2w_{ij}(x_{di} - x_{dj})^2/\sigma_d^3$.

2.3 Learning with Local and Global Consistency

The local and global consistency method [51] is similar to the harmonic function approach described above. It is different from the harmonic function approach in that it measures the smoothness of the classifying function in a different way. We first introduce some notations.

Given a point set $\mathcal{X} = \{x_1, \cdots, x_l, x_{l+1}, \cdots, x_n\} \subset \mathbb{R}^m$ and a label set $\mathcal{L} = \{1, \cdots, c\}$, the first l points $x_i (i \leq l)$ are labeled as $y_i \in \mathcal{Y}$ and the remaining points $x_u (l+1 \leq u \leq n)$ are unlabeled. The goal is to predict the label of the unlabeled points. Let $\mathcal{F}$ denote the set of $n \times c$ matrices with nonnegative entries. The matrix $F = [F_1^\top, \cdots, F_n^\top]^\top \in \mathcal{F}$ corresponds to a classification on the dataset $\mathcal{X}$ by labeling each point x_i as a label $y_i = \text{argmax}_{j \leq c} F_{ij}$. In other words, each row of F can be viewed as the confidence score vector of assigning all class labels to one data point and we can understand F as a vectorial function $F : \mathcal{X} \rightarrow \mathbb{R}^c$ which assigns a vector F_i to each point x_i. Define a $n \times c$ matrix $Y \in \mathcal{F}$ with $Y_{ij} = 1$ if x_i is labeled as $y_i = j$ and $Y_{ij} = 0$ otherwise. Y is consistent with the initial labels of the labeled data.

2.3.1 Description of the Method

The method can described in the following steps: first, we compute the weight matrix W defined as $W_{ij} = \exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)$ if $i \neq j$ and $W_{ii} = 0$; second, we construct the matrix $S = D^{-1/2} W D_{-1/2}$ in which D is diagonal matrix with its (i,i)-element equal to the sum of the i-th row of W; third, we conduct the iteration $F(t+1) = \alpha S F(t) + (1-\alpha) Y$ until it converges where α is a parameter in $(0,1)$; finally, each point x_i can be labeled as $y_i = \arg\max_{j \leq c} F_{ij}^*$.

This method can be easily understood from the propagation point of view. We first define the weight matrix W which capture the manifold structure of the dataset $\mathcal{D}$. The graph $G = (V, E)$ for propagation can be defined on $\mathcal{D}$, where the vertex set V includes all points from $\mathcal{D}$ and the edges E are weighted by W. W is normalized symmetrically for the convergence of the iteration. In the iteration step, the labels of the labeled data are repeatedly propagated over the graph G until convergence. It is not too hard to see that each point receives the information propagated from its neighbors and at the same time retains its initial information. The self-reinforcement is avoided since the diagonal elements of the weight matrix W are set to zero. After the propagation ends, we can label the unlabeled

data using F^*.

2.3.2 The Regularization Framework

Based on the above algorithm, a regularization framework can be developed as follows:

$$\mathcal{Q}(F) = \frac{1}{2}\left(\sum_{i,j=1}^{n} W_i \left\| \frac{1}{\sqrt{D_{ii}}}F_i - \frac{1}{\sqrt{D_{jj}}}F_j \right\|^2 + \mu \sum_{i=1}^{n} \|F_i - Y_i\|^2\right) \tag{2.10}$$

where $\mu > 0$ is the regularization parameter. Then the classifying function is

$$F^* = \text{argmin}_{F \in \mathcal{F}} \mathcal{Q}(F) \tag{2.11}$$

The first term of the right-hand side in Equation 2.10 is the *smoothness constant*, which means that a good classifying function should not change too much between nearby points. The second term is *fitting constraint*, which means a good classifying function should not change too much from the initial label assignment. The trade-off between two terms is captured by μ.

F^* can be solved by differentiating $\mathcal{Q}(F)$ with respect to F,

$$\frac{\partial \mathcal{Q}}{\partial F}|_{F=F^*} = F^* - SF^* + \mu(F^* - Y) = 0$$

$$F^* - \frac{1}{1+\mu}SF^* - \frac{\mu}{1+\mu}Y = 0$$

Two new variables are introduced, $\alpha = \frac{1}{1+\mu}$, and $\beta = \frac{\mu}{1+\mu}$. Note that $\alpha + \beta = 1$. Then

$$(I - \alpha S)F^* = \beta Y$$

Since $I - \alpha S$ is invertible, the solution is

$$F^* = \beta(I - \alpha S)^{-1} Y.$$

2.4 Summary

Label propagation is an important technique in machine learning and has shown the promising performance in many applications such as classification and ranking problems. The key idea behind label propagation is to propagate the labels of the labeled data to the unlabeled data over the graph constructed from the affinity matrix based on the data and then make predictions accordingly. Despite the various motivations behind different methods such as graph-cut based approaches [23, 4, 5], gaussian processes based approach [30], gaussian random fields and harmonic function based approach [57, 54, 56] and so on, most label propagation assumes the consistency of the resulting labels, which means that the data examples close to each other should have similar labels. The difference among various label propagation methods lie in the way to model the consistency of the data and propagate the labels. In addition to the overview of different algorithms for label propagation, we also presented in detail two most popular algorithms for label propagation including the gaussian random field and harmonic function based approach, and the local and global consistency approach due to their resemblance to our work in the following chapters.

Chapter 3

Relation Propagation:

A Generalized Framework

Despite the extensive study in label propagation, most previous work assumes that the similarity graph consists of a single type of object, namely documents, and the propagation is only conducted among objects of the same type. However, there are scenarios in many applications that involve multiple types of objects, and the label information needs to be propagated not only among objects of the same type, but also between objects of different types. To illustrate this issue, consider the multi-label classification problem. To directly employ the label propagation approach, we will first decompose the multi-label classification problem into a number of binary classification problems, and then the labels of each binary class is propagated through a similarity matrix. The problem with this approach is that it is incapable of exploring the correlation information among different classes, which is often extremely useful when the number of training documents is small. To see this, consider a five-class classification problem. Assume that for an unlabeled document $\mathbf{d}$, its confidence scores for the five classes after the binary class propagation are 0.4, 0.4, 0.4, 0.3, and 0.3. Without using the class correlation information, it is equally likely to assign the document $\mathbf{d}$ to category $1, 2$, and 3. However, if both category 4 and 5 are highly correlated

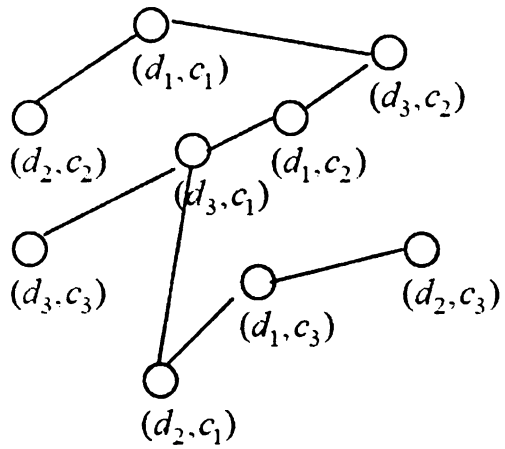


Figure 3.1: An Example of the Connected Graph for Relation Propagation

with category 3 and meanwhile are orthogonal to category 1 and 2, we would expect that category 3 is more likely to be the right class for document **d** than category 1 and 2.

We can generalize the above example into a propagation framework for multiple types of objects. In particular, we treat the documents and the categories as two different types of objects. Instead of propagating the labeling information among documents independently for each category, we propose to propagate the *relationship* between documents and categories. More specifically, we construct a weighted graph as follows: each node $O_{(i,j)} = (d_i, c_j)$ in the graph represents the relationship that document $\mathbf{d}_i$ belongs to the category c_j; any two nodes $O_{(i,j)}$ and $O_{(k,l)}$ in the graph are connected by an edge whose weight reflects the correlation between the two corresponding relationships. In other words, a large similarity between node $O_{(i,j)}$ and $O_{(k,l)}$ indicates that document $\mathbf{d}_i$ is likely to be assigned to class c_i if document $\mathbf{d}_k$ belongs to class c_l, and vice versa. It is important to note that class c_k and c_l can be different in the above propagation scheme. Figure 3.1 illustrates an example of the graph for the relation propagation with three documents and three categories. To distinguish from the previous work on label propagation, we refer to the proposed framework as "**Relation Propagation**", or **RP** for short.

We would like to emphasize that the relation propagation framework can be applied to

many applications in addition to the multi-label learning problems described above. For instance, it can be applied to collaborative filtering by viewing users and items as two different objects, and the relationship $O_{(i,j)} = (u_i, t_j)$ represents the rating of item t_j by user u_i. Using the framework of relation propagation, we will be able to propagate the rating information among different users and different items simultaneously. In particular, it allows us to infer the rating of user u_1 on item t_1 given the rating of user u_2 on a different item t_2 if items t_1 and t_2 share similar characteristics. This property can be extremely useful to alleviate the sparse data problem, which has been an critical issue in collaborative filtering [7].

3.1 A Graph-based Framework

To facilitate our discussion, we first describe the framework of relation propagation for two types of objects, followed by the generalization to multiple types of objects. We then present the efficient implementation of applying the proposed framework to multi-label learning and collaborative filtering.

Let $\mathcal{A} = (a_1, a_2, \ldots, a_n)$ and $\mathcal{B} = (b_1, b_2, \ldots, b_m)$ denote the two types of objects. Let $f(a_i, b_j) : \mathcal{A} \times \mathcal{B} \rightarrow \mathbf{R}$ denote the relation function between an object of type $\mathcal{A}$ and an object of type $\mathcal{B}$. Let $\mathbf{y}$ denote the vector of size mn whose element $y_{(i,j)}$ corresponds to the relation $f(a_i, b_j)$. Note that we use a double index (i, j) to refer an element in vector $\mathbf{y}$. This convention will be used throughout the entire paper. For the convenience of discussion, we assume that the first N_l elements in vector $\mathbf{y}$, denoted by $\mathbf{y}_l$, are the labeled relations, and the remaining $N_u = mn - N_l$ elements in $\mathbf{y}$, denoted by $\mathbf{y}_u$, are the unlabeled relations that need to be predicted. Finally, let $S^{\mathcal{A}} = [S^{\mathcal{A}}_{i,j}]_{n \times n}$ and $S^{\mathcal{B}} = [S^{\mathcal{B}}_{i,j}]_{m \times m}$ denote the matrices of similarities among the objects of type $\mathcal{A}$ and among the objects of type $\mathcal{B}$, respectively.

In order to incorporate the similarity information $S^{\mathcal{A}}$ and $S^{\mathcal{B}}$ into the propagation

scheme, we modify the energy function in Equation (2.4) into the following expression:

$$E_m = \sum_{i,k=1}^{n} \sum_{j,l=1}^{m} S_{i,k}^{\mathcal{A}} S_{j,l}^{\mathcal{B}} (y_{(i,j)} - y_{(k,l)})^2 \tag{3.1}$$

In the above, we introduce the weight $S_{i,k}^{\mathcal{A}} S_{j,l}^{\mathcal{B}}$, the product of the similarity measurements for the two types of objects, to weigh the difference between the two relations $y_{(i,j)}$ and $y_{(k,l)}$. Hence, to minimize the energy function in Equation (3.1), two relations will be enforced to have similar values when they share similar input objects in both type $\mathcal{A}$ and type $\mathcal{B}$. We then simplify the expression for energy E_m by using the matrix notation, i.e.,

$$E_m = \mathbf{y}^\top L^{\mathcal{A},\mathcal{B}} \mathbf{y} \tag{3.2}$$

where

$$\begin{aligned} L^{\mathcal{A},\mathcal{B}} &= D^{\mathcal{A},\mathcal{B}} - S^{\mathcal{A},\mathcal{B}} \\ S^{\mathcal{A},\mathcal{B}} &= S^{\mathcal{A}} \bigotimes S^{\mathcal{B}} \end{aligned}$$

where operator $\bigotimes$ stands for the direct product of matrices. $D^{\mathcal{A},\mathcal{B}}$ in the above expression is defined as a diagonal matrix whose diagonal element is defined as

$$\begin{aligned} D_{(i,j),(i,j)}^{\mathcal{A},\mathcal{B}} &= \sum_{k=1}^{n} \sum_{l=1}^{m} S_{(i,j),(k,l)}^{\mathcal{A},\mathcal{B}} \\ &= \left(\sum_{k=1}^{n} S_{i,k}^{\mathcal{A}} \right) \left(\sum_{l=1}^{m} S_{j,l}^{\mathcal{B}} \right) \\ &= D_i^{\mathcal{A}} D_j^{\mathcal{B}} \end{aligned} \tag{3.3}$$

Finally, similar to the solution in Equation (2.5), the optimal solution for $\mathbf{y}_u$ that minimizes

the energy function in Equation (3.2) is

$$\mathbf{y}_u = \left[L_{u,u}^{\mathcal{A},\mathcal{B}}\right]^{-1} S_{u,l}^{\mathcal{A},\mathcal{B}} \mathbf{y}_l \tag{3.4}$$

As the further improvement, we introduce the normalized similarity matrices, and the normalized graph Laplacian:

$$\bar{S}^{\mathcal{A}} = \left(D^{\mathcal{A}}\right)^{-1/2} S^{\mathcal{A}} \left(D^{\mathcal{A}}\right)^{-1/2} \tag{3.5}$$

$$\bar{S}^{\mathcal{B}} = \left(D^{\mathcal{B}}\right)^{-1/2} S^{\mathcal{B}} \left(D^{\mathcal{B}}\right)^{-1/2} \tag{3.6}$$

$$\bar{S}^{\mathcal{A},\mathcal{B}} = \left(D^{\mathcal{A},\mathcal{B}}\right)^{-1/2} S^{\mathcal{A},\mathcal{B}} \left(D^{\mathcal{A},\mathcal{B}}\right)^{-1/2} \tag{3.7}$$

$$\begin{aligned} \bar{L}^{\mathcal{A},\mathcal{B}} &= \left(D^{\mathcal{A},\mathcal{B}}\right)^{-1/2} L^{\mathcal{A},\mathcal{B}} \left(D^{\mathcal{A},\mathcal{B}}\right)^{-1/2} \\ &= I - \bar{S}^{\mathcal{A},\mathcal{B}} \end{aligned} \tag{3.8}$$

Notice that, according to the above definitions, we have

$$\bar{S}^{\mathcal{A},\mathcal{B}} = \bar{S}^{\mathcal{A}} \bigotimes \bar{S}^{\mathcal{B}} \tag{3.9}$$

This is because

$$\begin{aligned} \bar{S}_{(i,k),(j,l)}^{\mathcal{A},\mathcal{B}} &= \frac{S_{(i,k),(j,l)}^{\mathcal{A},\mathcal{B}}}{\sqrt{D_{(i,k),(i,k)}^{\mathcal{A},\mathcal{B}} D_{(j,l),(j,l)}^{\mathcal{A},\mathcal{B}}}} \\ &= \frac{S_{i,j}^{\mathcal{A}} S_{k,l}^{\mathcal{B}}}{\sqrt{D_i^{\mathcal{A}} D_k^{\mathcal{B}} D_j^{\mathcal{A}} D_l^{\mathcal{B}}}} \\ &= \frac{S_{i,j}^{\mathcal{A}}}{\sqrt{D_i^{\mathcal{B}} D_j^{\mathcal{A}}}} \frac{S_{i,j}^{\mathcal{B}}}{\sqrt{D_i^{\mathcal{A}} D_j^{\mathcal{B}}}} \bar{S}_{i,j}^{\mathcal{A}} \bar{S}_{k,l}^{\mathcal{B}} \end{aligned}$$

Replacing the graph Laplacian $L^{\mathcal{A},\mathcal{B}}$ and similarity matrix $S^{\mathcal{A},\mathcal{B}}$ in Equation (3.4) with the

normalized ones, we have $\mathbf{y}_u$ computed as:

$$\mathbf{y}_u = \left(I - \bar{S}_{u,u}^{\mathcal{A},\mathcal{B}}\right)^{-1} \bar{S}_{u,l}^{\mathcal{A},\mathcal{B}} \mathbf{y}_l \tag{3.10}$$

It is straightforward to extend the above formulism to the case of multiple types of objects. Let $\mathcal{T} = (\mathcal{O}^1, \mathcal{O}^2, \ldots, \mathcal{O}^t)$ denote the t different types of objects. Let $f : \mathcal{O}^1 \times \mathcal{O}^2 \ldots \times \mathcal{O}^t \rightarrow \mathbf{R}$ denote the relation function among t different types of objects. Let $\{S^k = [S^k_{i,j}]_{n_k \times n_k}, k = 1, 2, \ldots, t\}$ denote the similarity matrices for the t types of objects. We then have the energy function for the t types of objects written as:

$$E_m = \mathbf{y}^\top \left(\bigotimes_{k=1}^{t} S^k \right) \mathbf{y}$$

where each element in the vector $\mathbf{y}$, i.e., $y_{(o^1_{i_1}, o^2_{i_2}, \ldots, o^t_{i_t})}$, corresponds to the relation $f(o^1_{i_1}, o^2_{i_2}, \ldots, o^t_{i_t})$. A solution similar to the one in Equation (3.4) will minimize the above energy function.

3.2 Applications for Classification Problem

The relation propagation approach can be used for classification problem. This section will discuss its application on multi-label learning task. We will describe the multi-label learning model based on relation propagation and then present the empirical studies with different datasets.

3.2.1 Multi-Label Learning Model

It is straightforward to apply the framework of relation propagation to multi-label learning. Let's denote the collection of documents by $\mathcal{D} = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n)$, and the set of categories by $\mathcal{C} = (c_1, c_2, \ldots, c_m)$. Then, the relation $f(\mathbf{d}_i, c_j)$ is a binary function that outputs 1 when document $\mathbf{d}_i$ belong to the j-th category, and 0 otherwise. Let the normalized similarity

matrices of categories and documents denoted by $\bar{S}^{\mathcal{C}}$ and $\bar{S}^{\mathcal{D}}$, and the normalized graph Laplacian denoted by $\bar{L}^{\mathcal{D}}$.

Assume that the first n_l documents of $\mathcal{D}$ are the labeled examples, and the rest $n_u = n - n_l$ documents are the unlabeled examples. We then decompose the document similarity matrix $\bar{S}^{\mathcal{D}}$ as follows:

$$\bar{S}^{\mathcal{D}} = \begin{pmatrix} \bar{S}^{\mathcal{D}}_{l,l} & \bar{S}^{\mathcal{D}}_{l,u} \\ \bar{S}^{\mathcal{D}}_{u,l} & \bar{S}^{\mathcal{D}}_{u,u} \end{pmatrix}$$

where the subindexes "l" and "u" refer to the labeled documents and the unlabeled documents, respectively. Similarly, we can decompose the normalized matrix $\bar{S}^{\mathcal{D},\mathcal{C}}$ and $\bar{L}^{\mathcal{D},\mathcal{C}}$ as follows:

$$\begin{aligned} \bar{S}^{\mathcal{D},\mathcal{C}} &= \bar{S}^{\mathcal{D}} \bigotimes \bar{S}^{\mathcal{C}} = \begin{pmatrix} \bar{S}^{\mathcal{D}}_{l,l} \bigotimes \bar{S}^{\mathcal{C}} & \bar{S}^{\mathcal{D}}_{l,u} \bigotimes \bar{S}^{\mathcal{C}} \\ \bar{S}^{\mathcal{D}}_{u,l} \bigotimes \bar{S}^{\mathcal{C}} & \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}} \end{pmatrix} \\ \bar{L}^{\mathcal{D},\mathcal{C}} &= I - \bar{S}^{\mathcal{D},\mathcal{C}} = \begin{pmatrix} \bar{L}^{\mathcal{D}}_{l,l} & \bar{L}^{\mathcal{D}}_{l,u} \\ \bar{L}^{\mathcal{D}}_{u,l} & \bar{L}^{\mathcal{D}}_{u,u} \end{pmatrix} \\ &= \begin{pmatrix} I - \bar{S}^{\mathcal{D}}_{l,l} \bigotimes \bar{S}^{\mathcal{C}} & -\bar{S}^{\mathcal{D}}_{l,u} \bigotimes \bar{S}^{\mathcal{C}} \\ -\bar{S}^{\mathcal{D}}_{u,l} \bigotimes \bar{S}^{\mathcal{C}} & I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}} \end{pmatrix} \end{aligned}$$

Using the above expressions for $\bar{S}^{\mathcal{D},\mathcal{C}}_{u,l}$ and $\bar{L}^{\mathcal{D},\mathcal{C}}_{u,u}$, the solution in Equation (3.10) for multi-label learning can be expanded as:

$$\mathbf{y}_u = \left(I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}}\right)^{-1} \left(\bar{S}^{\mathcal{D}}_{u,l} \bigotimes \bar{S}^{\mathcal{C}}\right) \mathbf{y}_l \tag{3.11}$$

Directly computing the solution in the above expression could be computationally expensive. This is because the size of the matrix $I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}}$ is $O(mn \times nm)$, roughly the square of the product between the number of documents and the number of categories.

This matrix will be huge even when the number of documents and the number of categories are modest, thus making it unfeasible to store this matrix, not to mention computing the inverse of this matrix. For instance, in our experiment with multi-label learning, the number of documents is 1000 and the number of categories is 100. This results in matrix $I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}}$ of size $100,000 \times 100,000$, which is too large to be manipulated by most desktop computers. In the following, we present an efficient algorithm for applying relation propagation to multi-label learning.

The key idea of the efficient algorithm is to approximate the inverse of matrix $I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}}$ by its eigenvectors. To this end, we first present an important theorem regarding the eigenvectors of the direct product of matrices, which will serve as the basis for our efficient algorithm.

Theorem 1 *Let $\mathbf{v}_a$ and $\mathbf{v}_b$ are the eigenvectors of matrix A and B, and λ_a and λ_b are the corresponding eigenvalues. Then, $\mathbf{v}_a \bigotimes \mathbf{v}_b$ is an eigenvector of matrix $A \bigotimes B$, and $\lambda_a \lambda_b$ is the corresponding eigenvalue.*

Proof To prove the above theorem, we need to show the following equation holds

$$\left(A \bigotimes B\right)(\mathbf{v}_a \bigotimes \mathbf{v}_b) = \lambda_a \lambda_b (\mathbf{v}_a \bigotimes \mathbf{v}_b) \tag{3.12}$$

Notice that the left side of the equation in the above can be simplified as

$$\begin{aligned} \left(A \bigotimes B\right)(\mathbf{v}_a \bigotimes \mathbf{v}_b) &= (A\mathbf{v}_a) \bigotimes (B\mathbf{v}_b) \\ &= \lambda_a \lambda_b (\mathbf{v}_a \bigotimes \mathbf{v}_b) \end{aligned}$$

In the above, we use the following property of the matrix direct product:

$$\left(U \bigotimes V\right)\left(C \bigotimes D\right) = (UC) \bigotimes (UD) \tag{3.13}$$

Thus, the left hand side of Equation (3.12) is equal to its right hand side, and the theorem

is proved. Using Theorem 1, we can approximate the expression in (3.11) using the eigenvectors of matrices $\bar{S}^{\mathcal{D}}_{u,u}$ and $\bar{S}^{\mathcal{C}}$. Let the top eigenvalues and eigenvectors of $\bar{S}^{\mathcal{D}}_{u,u}$ denoted by $\{(\lambda^{\mathcal{D}}_i, \mathbf{v}^{\mathcal{D}}_i), i = 1, 2, \ldots, K_{\mathcal{D}}\}$, and the top eigenvalues and eigenvectors of $\bar{S}^{\mathcal{C}}$ denoted by $\{(\lambda^{\mathcal{C}}_k, \mathbf{v}^{\mathcal{C}}_k), k = 1, 2, \ldots, K_{\mathcal{C}}\}$. Then, the eigenvectors and eigenvalues of $I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes S^{C}$ can be expressed as follows:

$$\begin{aligned} \lambda_{(i,k)} &= 1 - \lambda^{\mathcal{D}}_i \lambda^{\mathcal{C}}_k, \\ \mathbf{v}_{(i,k)} &= \mathbf{v}^{\mathcal{D}}_i \bigotimes \mathbf{v}^{\mathcal{C}}_k,\ i = 1, \ldots, K_{\mathcal{D}},\ k = 1, \ldots, K_{\mathcal{C}} \end{aligned}$$

Therefore, the inverse of $I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}}$ is calculated as:

$$\begin{aligned} &(I - S^{D}_{u,u} \bigotimes S^{C})^{-1} \approx \\ &\quad \sum_{i=1}^{K_{\mathcal{D}}} \sum_{k=1}^{K_{\mathcal{C}}} \frac{1}{1 - \lambda^{\mathcal{D}}_i \lambda^{\mathcal{C}}_k} \left(\mathbf{v}^{\mathcal{D}}_i \bigotimes \mathbf{v}^{\mathcal{C}}_k\right) \left(\mathbf{v}^{\mathcal{D}}_i \bigotimes \mathbf{v}^{\mathcal{C}}_k\right)^{\top} \end{aligned}$$

Then, the product $(I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}})^{-1} (\bar{S}^{\mathcal{D}}_{u,l} \bigotimes \bar{S}^{\mathcal{C}})$ is simplified as

$$\begin{aligned} &(I - \bar{S}^{\mathcal{D}}_{u.u} \bigotimes \bar{S}^{\mathcal{C}})^{-1} (\bar{S}^{\mathcal{D}}_{u.l} \bigotimes \bar{S}^{\mathcal{C}}) \\ &= \sum_{i=1}^{K_{\mathcal{D}}} \sum_{k=1}^{K_{\mathcal{C}}} \frac{1}{1 - \lambda^{\mathcal{D}}_i \lambda^{\mathcal{C}}_k} \left(\mathbf{v}^{\mathcal{D}}_i \bigotimes \mathbf{v}^{\mathcal{C}}_k\right) \left(\mathbf{v}^{\mathcal{D}}_i \bigotimes \mathbf{v}^{\mathcal{C}}_k\right)^{\top} \left(\bar{S}^{\mathcal{D}}_{u.l} \bigotimes \bar{S}^{\mathcal{C}}\right) \\ &= \sum_{i=1}^{K_{\mathcal{D}}} \sum_{k=1}^{K_{\mathcal{C}}} \frac{\lambda^{\mathcal{C}}_k}{1 - \lambda^{\mathcal{D}}_i \lambda^{\mathcal{C}}_k} \left(\mathbf{v}^{\mathcal{D}}_i \bigotimes \mathbf{v}^{\mathcal{C}}_k\right) \left(\left[\mathbf{v}^{\mathcal{D}}_i\right]^{\top} \bar{S}^{\mathcal{D}}_{u.l}\right) \bigotimes \left[\mathbf{v}^{\mathcal{C}}_k\right]^{\top} \end{aligned}$$

In the last of the derivation, we again use the property of direct product in Equation (3.13), and the property of eigenvectors, namely $\bar{S}^{\mathcal{C}} \mathbf{v}^{\mathcal{C}}_k \lambda^{\mathcal{C}}_k \mathbf{v}^{\mathcal{C}}_k$. Using the above expression for $(I - \bar{S}^{\mathcal{D}}_{u.u} \bigotimes \bar{S}^{\mathcal{C}})^{-1} (\bar{S}^{\mathcal{D}}_{u.l} \bigotimes \bar{S}^{\mathcal{C}})$, the predict class labels $\mathbf{y}_u$ in (3.11) is then rewritten as:

$$\mathbf{y}_u = \sum_{i=1}^{K_{\mathcal{D}}} \sum_{k=1}^{K_{\mathcal{C}}} \frac{\lambda^{\mathcal{C}}_k}{1 - \lambda^{\mathcal{D}}_i \lambda^{\mathcal{C}}_k} \times \left(\mathbf{v}^{\mathcal{D}}_i \bigotimes \mathbf{v}^{\mathcal{C}}_k\right) \left(\left[\mathbf{v}^{\mathcal{D}}_i\right]^{\top} \bar{S}^{\mathcal{D}}_{u.l}\right) \bigotimes \left[\mathbf{v}^{\mathcal{C}}_k\right]^{\top} \mathbf{y}_l \tag{3.14}$$

Comparing to the original expression for $\mathbf{y}_u$ in Equation (3.11), the above expression is

computationally efficient because it no longer needs to compute the inverse of matrix $I - \bar{S}^{\mathcal{D}}_{u,u} \bigotimes \bar{S}^{\mathcal{C}}$ explicitly. Figure 3.2 summarizes the procedure for efficiently predicting the class labels for unlabeled data using the framework of relation propagation. A careful examination of the algorithm in Figure 3.2 indicates that our algorithm does not need to compute any matrix of size $O(nm \times nm)$. In fact, during the iterations, only vectors of size $O(mn)$ are computed, and no any intermediate matrix is calculated. Hence, this algorithm is efficient not only in computation but also in memory storage.

Input:
- The labels of the training documents $\mathbf{y}_l$;
- $S^{\mathcal{D}}$ an $S^{\mathcal{C}}$: similarity matrices for documents and categories;
- $K_{\mathcal{D}}$ and $K_{\mathcal{C}}$: the number of required eigenvectors for the similarity matrices of documents and categories.

Output: The class labels for the unlabeled documents $\mathbf{y}_u$

1. Computing the eigen space
- Compute normalized similarity matrices $\bar{S}^{\mathcal{D}}$ and $\bar{S}^{\mathcal{C}}$ as in Equation (3.5) and (3.6)
- Compute the top $K_{\mathcal{D}}$ eigenvalues and eigenvectors of $\bar{S}^{\mathcal{D}}$, i.e., $(\lambda_i^{\mathcal{D}}, \mathbf{v}_i^{\mathcal{D}}), i = 1, 2, \ldots, K_{\mathcal{D}}$
- Compute the top $K_{\mathcal{C}}$ eigenvalues and eigenvectors of $\bar{S}^{\mathcal{C}}$, i.e., $(\lambda_k^{\mathcal{C}}, \mathbf{v}_k^{\mathcal{C}}), k = 1, 2, \ldots, K_{\mathcal{C}}$

2. Initialization: $\mathbf{y}_u = \mathbf{0}_{n_u \times m}$

3. For $i = 1, 2, \ldots, K_{\mathcal{D}}$, and $k = 1, 2, \ldots, K_{\mathcal{C}}$
- Compute $\mathbf{a} = \left[\bar{S}^{\mathcal{D}}_{u,l}\right]^{\top} \mathbf{v}_i^{\mathcal{D}}$
- Compute $\mathbf{b} = \left[\mathbf{a} \bigotimes \mathbf{v}_k^{\mathcal{C}}\right]^{\top} \mathbf{y}_l$
- Compute $\mathbf{c} = \left(\mathbf{v}_i^{\mathcal{D}} \bigotimes \mathbf{v}_k^{\mathcal{C}}\right) \mathbf{b}$
- Update $\mathbf{y}_u = \mathbf{y}_u + \lambda_k^{\mathcal{C}} \mathbf{c} / (1 - \lambda_i^{\mathcal{D}} \lambda_k^{\mathcal{C}})$

Figure 3.2: An Efficient Algorithm for Multi-label Learning using the Relation Propagation

3.2.2 Case Study: Image Categorization

The goal of the experiments is to evaluate the effectiveness of the proposed framework of relation propagation for multi-label learning. In particular, we will address the following research questions in this empirical study:

- *Will the relation propagation framework be effective for multi-label learning*? In particular, we will examine whether the incorporation of class correlation and the unlabeled documents will improve the classification performance. To this end, we compare the proposed algorithm to two graph-based approaches, the label propagation approach based on the harmonic function [57] and the spectral graph transducer (SGT) [24].
- *How sensitive is the proposed algorithm for multi-label learning to the setup of parameters*? As indicated in Figure 3.2, the input to the proposed algorithm includes the two similarity matrices, $S^{\mathcal{D}}$ and $S^{\mathcal{C}}$, and the number of required eigenvectors, $K_{\mathcal{D}}$ and $K_{\mathcal{C}}$. In this experiment, we will vary these input variables to examine their impact on the classification performance of the proposed algorithm.

Dataset

We evaluate the proposed algorithm for multi-label learning on the Eurovision ST. Andrews photographic collection (ESTA)[1], which is provided by Text and Content-Based Cross Language Image Retrieval (ImageCLEF)[2]. This collection contatins 28133 images that belong to 999 pre-defined categories. Each image is also described a short textual document. The averaged length is about 50 words per document. In our experiment, we randomly selected 100 categories and textual description of 1000 images as our testbed. For the selected testbed, the number of documents for each category varies from 2 to around 600, and the average number of documents per category is around 72. Meanwhile, the number of categories per document varies from 6 to 11, and the average number of categories per document is around 7.

[1] http://ir.shef.ac.uk/imageclef/2004/stand.html
[2] http://ir.shef.ac.uk/imageclef/

Evaluation Methodology

Similar to the previous studies on multi-label text categorization, we evaluate the proposed algorithm using both the F1 measure across documents and the F1 measure across categories. The first metric, referred as "micro-average", is calculated by first computing the precision and the recall of class labels for each document, and then taking the average of the precision and the recall over all the documents in the test set, and finally computing F1 measure based on the average precision and average recall. The second metric, referred as "macro-average", is calculated by first computing the average precision and recall for documents within each category, and then taking the average of the precision and the recall over all the categories, and finally computing the F1 measure based on the average precision and average recall. Since the micro-average is usually dominated by the popular categories while the macro-average is usually decided by the rare categories, by using both metrics, we will be able to obtain the comprehensive view of the classification performance for the proposed algorithm.

Furthermore, similar to the label ranking algorithms for multi-label learning, the proposed algorithm only estimates the confidence scores of class labels for the unlabeled documents. An additional procedure is needed to decide the set of class labels for each unlabeled document. To obtain a more comprehensive view of the classification performance for the proposed algorithm, for each document, we first rank its class labels in the descending order of the estimated confidence scores. We then compute the F1 measure based on average precision and recall at first 20 ranks. This is similar to the analysis based on Receiver Operating Characteristic (ROC) Curves [35].

Finally, since the focus of our studies is on semi-supervised learning, 2%, 5% and 10% of documents are randomly chosen as training set and the rest of the documents are used for testing. Each experiment is conducted for 10 times, and the average precision and recall across 10 trials are used to compute the final F1 measure.

Parameter Setup and Baseline Models

As indicated in Figure 3.2, there are four inputs to the proposed algorithm for multi-label learning: $S^{\mathcal{D}}$, $S^{\mathcal{C}}$, $K_{\mathcal{D}}$, and $K_{\mathcal{C}}$. To obtain the document similarity matrix $S^{\mathcal{D}}$, the SMART information retrieval system [39] is first used for preprocessing the documents to generate the weighted term vectors. The pre-processing includes removing stopwords, stemming, and the Okapi term weighting [37]. Then, a RBF kernel function $f(\mathbf{t}_i, \mathbf{t}_j) = \exp(-\gamma \|\mathbf{t}_i - \mathbf{t}_j\|_2^2 / \bar{d})$ is used to calculate the document similarity based on their weighted term vectors $\mathbf{t}_i$ and $\mathbf{t}_j$, where $\bar{d}$ is average distance between any two documents, and γ is the scaling factor. In the experiment, we will examine impact of parameter γ on the classification performance of the proposed algorithm.

The category similarity matrix $S^{\mathcal{C}}$ is obtained by first representing each category as a binary vector in the space of documents, and using the cosine similarity between the vectors of categories as the similarity measurement of categories. Since the number of categories is only 100, we set $K_{\mathcal{C}} = 100$ for all experiments.

We compare our approach for multi-label learning to two state-of-art classification algorithms: the label propagation (LP) approach that is based on the harmonic function in [57], and the spectral graph transducer (SGT) [24]. Both these two approaches are semi-supervised learning algorithms. In particular, they explore the distribution of unlabeled documents for predicting their class labels.

SGT is a semi-supervised version of ratio-cut algorithm originally proposed for unsupervised learning [23]. The objective function incorporates a quadratic penalty on labeled data in addition to minimize the graph cut as

$$\min_{\mathbf{f}} = \mathbf{f}^\top \mathbf{L}\mathbf{f} + c(\mathbf{f} - \mathbf{r})^\top \mathbf{C}(\mathbf{f} - \mathbf{r})$$

where the vector f is a label probability vector, and the vector r is defined as

$$r_i = \begin{cases} \hat{r}_+ & x_i \text{ is a positive example} \\ \hat{r}_- & x_i \text{ is a negative example} \\ 0 & x_i \text{ is unlabeled} \end{cases}$$

The matrix $\mathbf{C} = diag(c_1, c_2, \cdots, c_n)$ is a diagnal cost matrix allowing for different misclassification cost for each data example. The trade-off between graph cut value and training error penalty is balanced through the constant c.

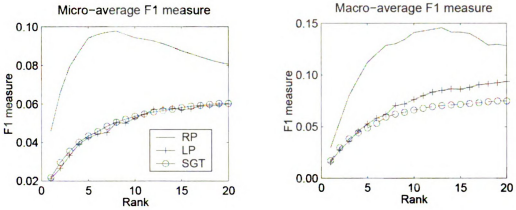
Previous studies have shown that these two approaches are able to outperform a number of comparative semi-supervised learning algorithm, such as Transductive Support Vector Machine (TSVM) [24]. However, one drawback with these two approaches is that none of them is able to explore the correlation among different classes. By comparing to them, we are able to examine if the proposed algorithm is able to improve the classification performance by incorporating the class correlation information. The implementation of Spectral Graph Transducer used in our study is download from (http://svmlight.joachims.org/). For easy reference, we refer to the proposed algorithm for multi-label learning as "RP". Finally, all the three baseline models use the same document similarities as the proposed algorithm.

Experiment (I): Comparison to Baseline Models

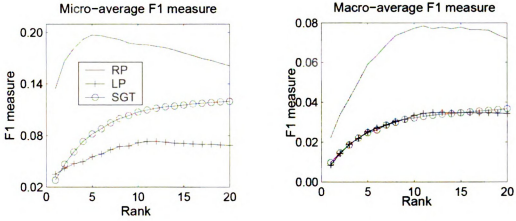
Figure 3.3 shows the classification results when 2%, 5% and 10% of the documents are used for training. The left panel in each figure shows the F1 measure of micro-average, and the right panel shows the F1 measure of macro-average. In this experiment, we set the scaling factor γ to be 1, $K_{\mathcal{D}} = 20$ and $K_{\mathcal{C}} = 100$.

First, we observe that as the size of training set increases, both the micro-average and macro-average F1 measures are improved for all three algorithms. This is expected because more training data will provide more information and thus improve the performance. Second, we observe that the spectral graph transducer is able to outperform the label prop-

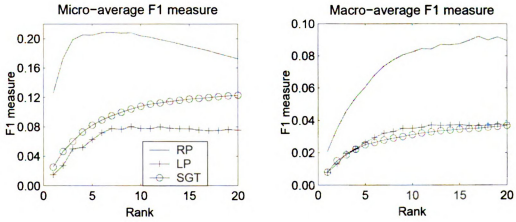
Figure 3.3: Classification F1 Results comparison using Relation Propagation



(a) Percentage of Training Data = 2%



(b) Percentage of Training Data = 5%



(c) Percentage of Training Data = 10%

agation algorithm noticeably when more than 2% of documents are used for training. This is consistent with the results of the previous study in [24]. Third, compared to label propagation and spectral graph transducer, we observe that the relation propagation algorithm is able to achieve significantly better classification accuracy for all the cases in terms of both micro-F1 and macro-F1 measurement. This result indicates that exploring the relationship among different types of objects does help improve the classification performance.

Experiment (II): Sensitivity Analysis

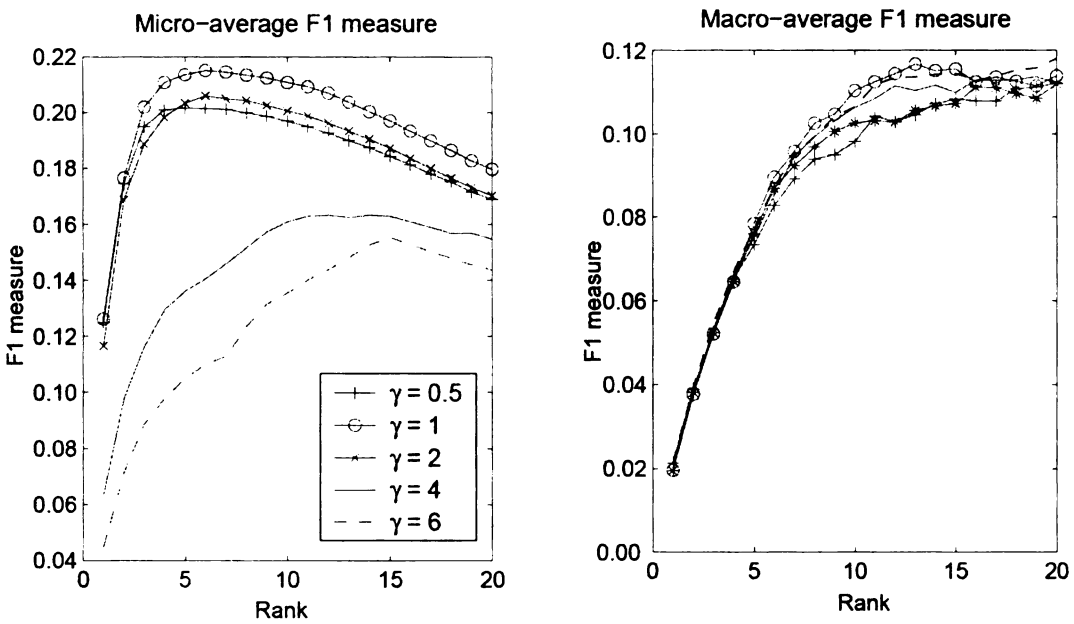


Figure 3.4: Classification Results of the Relation Propagation using Different γs. $K_{\mathcal{D}} = 20$

In this experiment, we examine how the scaling parameter γ and the number of eigenvectors $K_{\mathcal{D}}$ will affect the performance of the proposed relation propagation algorithm. The number of eigenvectors for categories $K_{\mathcal{C}}$ is set to be 100.

In the first experiment, we fix $K_{\mathcal{D}} = 20$ and vary γ from 0.5 to 1, 2, 4 to 6. Figure 3.4 shows the micro-averaged F1 and the macro-averaged F1 of the proposed algorithm for different γ. We observe that the classification performance of the proposed algorithm is not impacted too much when γ is set to be 0.5, 1, and 2. A significant degradation

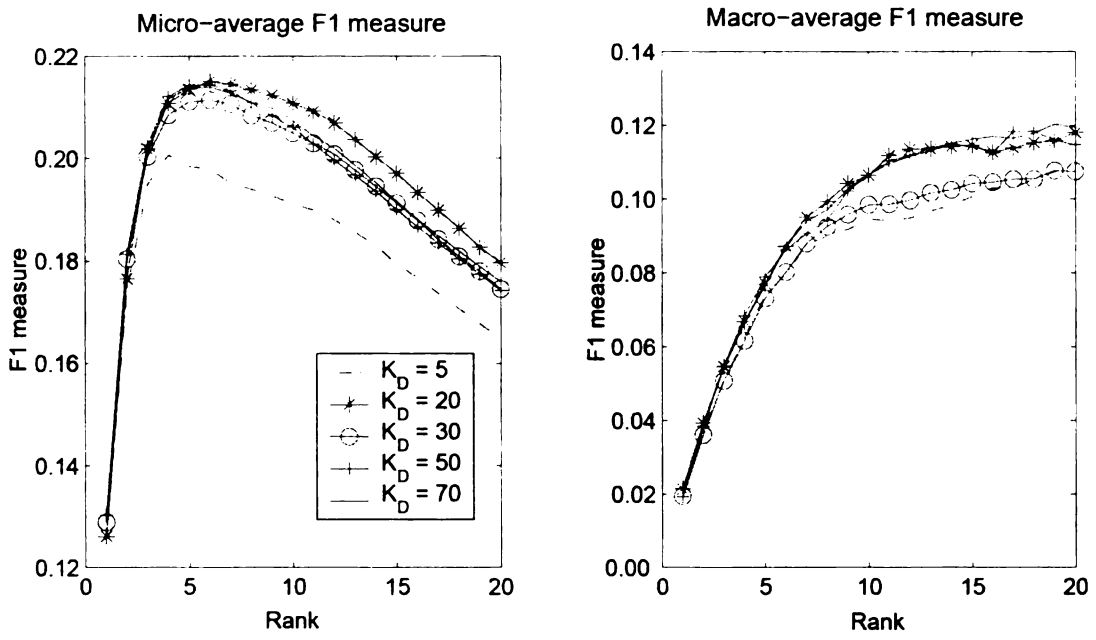


Figure 3.5: Classification Results of the Relation Propagation using Different $K_{\mathcal{D}}$s. $\gamma = 1$

is observed when γ is increased to 4 and 6. This is because, when the scaling γ is too large, the similarity between any two documents based on the RBF function will almost be zero, thus making it impossible to explore the unlabeled data. This is consistent with the observation in [57], i.e., only when it is possible to explore the correlation among unlabeled data only when the scaling parameter γ is small.

In the second experiment, we fix $\gamma = 1$ and vary the number of eigenvectors $K_{\mathcal{D}}$ from 5 to 20, 30, 50, and 70. Figure 3.5 shows the results for different $K_{\mathcal{D}}$. We observe that the classification performance was very stable for $K_{\mathcal{D}} \leq 20$ for both the micro-average F1 measure and macro-average F1 measure. However, the performance is significantly degraded when $K_{\mathcal{D}} = 5$ for micro-average F1 measure. Overall, the macro-average F1 measure is not affected significantly by the number of eigenvectors. Generally speaking, when $K_{\mathcal{D}} = 20$, the algorithm has the best performance.

3.3 Applications for Ranking Problem

The relation propagation model can be easily applied to ranking problems. Since the model generates the confidence scores for the objects in question, it is straightforward to rank the objects according to their confidence scores. We will use the collaborative filtering application as an example and describe how we can apply the relation propagation approach to the ranking problem. Empirical studies will be discussed after the model description.

3.3.1 Ranking Learning Model for Collaborative Filtering

The goal of collaborative filtering is to predict the preferences or interests of a particular user based on the available votes of training users, which sometimes is called community data. Many methods have been proposed for collaborative filtering in the past years.

Most methods assume that users with similar interests tend to rate items similarly. Thus, the similarity between users in their rating patterns becomes the key to most of the collaborative filtering methods. The user similarity can be used either for identifying a subset of training users that share similar interests with a test user, or for clustering a large number of users into a small number of user classes.

One of the key challenges to collaborative filtering is the sparse data problem, namely each user only provides a limited number of ratings. This is often the case in the real world applications given few users are willing to spend time on rating a large number of items. The sparse data problem often results in the unreliable estimation of the user similarity, which can significantly degrade the performance of collaborative filtering. This is because two users with similar interests may not share any commonly rated items, and as a result, have a zero user similarity. Many studies have been devoted to the sparse data problem in collaborative filtering [49, 6]. One strategy [19, 34] is to cluster training users into a small number of classes. Instead of identifying the individual training users that are similar to a test user, we identify the class of training users that fits in best with the test user. Another

strategy [57] is to fill out the ratings of the unrated items for the training users. This can be done either by propagating the rating of one item by one training user to the rating of another item by the same user if these two items are similar, or by propagating the rating of one item by one training user to the rating of the same item by another user if these two users share similar ratings for a number of items.

Our **Relation Propagation** model can be applied to address the sparse data problem in collaborative filtering. The key observation behind this work is that most of the previous work on collaborative filtering addresses the sparse data problem by exploring the user similarity and/or the item similarity separately. As a result, it only allows the rating information of the *same* item to be propagated among different users, or the rating information of the *same* user to be propagated among different items. This limitation will prevent the system from answering the question such as, if user y_1 rate item x_1 as 5, what is the expected rating of item x_2 by user y_2 provided that user y_1 and y_2 are similar in their preference of items, and in the meantime item x_1 and x_2 are similar. The relation propagation approach for collaborative filtering can address this problem by allowing the rating information to be propagated among different users and different items *simultaneously*.

To apply the framework, we first construct a weighted graph as follows: each node $v_{(i,j)} = (u_i, o_j)$ in the graph represents the rating of item o_j by user u_i; any two nodes $v_{(i,j)}$ and $v_{(k,l)}$ in the graph are connected by an edge whose weight reflects the correlation between the two corresponding rating relationships: a large weight indicates that user u_i will give a similar rating for item o_j as the rating by user u_k for item o_l, and vice versa.

The rest of this section is arranged as follows: We first present how the label propagation can be applied on collaborative filtering; we then introduce the relation propagation model for collaborative filtering, and the related algorithm.

To better motivate the proposed framework for collaborative filtering, we will start with the description of the label propagation method for collaborative filtering. We will then describe the relation propagation framework, which can be viewed as a generalization of

the label propagation method in [57]. Before getting to the details, we will first introduce the notations that will be used throughout this paper.

Let $\mathcal{U} = (u_1, u_2, \ldots, u_n)$ denote all users, where n is the number of users including both the training users and the test users. Let the similarities of users denoted by the matrix $S^u = [S^u_{i,j}]_{n \times n}$, where each element $S_{i,j}$ represents the similarity between two users in their rating patterns. Let the collection of items denoted by $\mathcal{O} = (o_1, o_2, \ldots, o_m)$ where m is the number of items. Let the similarity of items denoted by the matrix $S^o = [S^o_{i,j}]_{m \times m}$ where element $S^o_{i,j}$ represents the similarity between two items in their descriptive features. In the case of movie recommendation, each item corresponds to a different movie, and is described by a set of movie categories. Let the ratings of items $\mathcal{O}$ by users $\mathcal{U}$ denoted by the matrix $R = [R_{i,j}]_{n \times m}$. Each element $R_{i,j} \in \{0, 1, 2, \ldots, r\}$ represents the rating of the j-th item by the i-th user. It takes value between 1 and r when the j-th item is rated by the i-th user, and 0 when it is not. We also write matrix R as $R = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m)$ and $R = (\tilde{\mathbf{r}}_1, \tilde{\mathbf{r}}_2, \ldots, \tilde{\mathbf{r}}_n)^\top$, where each vector $\mathbf{r}_i$ represents the ratings of the i-th item by all the users, and $\tilde{\mathbf{r}}_i$ represents the ratings of all the items by the i-th user. Given the above information, the goal of collaborative filtering is to predict the rating of a given item by the test user.

In order to apply the label propagation method to collaborative filtering, we will erase the difference between the training users and the test users. This is because predicting the rating of items by the test users is equivalent to predicting the rating of the unrated items by the training users. So, the question that label propagation method will address is to predict the ratings of a given item o_k for *all* the users. For the sake of simplicity, let's assume that only the first n_l users among $\mathcal{U}$ rate the item o_k, and our goal is to estimate the ratings of item o_k for the remaining $n_u = n - n_l$ users. For the convenience of the presentation, we refer to the first n_l users as the "labeled" users and the remaining n_u users as the "unlabeled" users. Following the idea of the label propagation [57], we search for

the rating vector $\mathbf{r}_k$ that minimizes the following energy function:

$$E_k^l = \sum_{i,j=1}^{n} S_{i,j}^u (R_{i,k} - R_{j,k})^2 = \mathbf{r}_k^\top L^u \mathbf{r}_k \tag{3.15}$$

where matrix $L^u = [L_{i,j}^u]_{n \times n}$ is the graph Laplacian for the similarity matrix S^u. It is defined as $L^u = D^u - S^u$ where $D^u = \text{diag}(D_1^u, D_2^u, \ldots, D_n^u)$ and $D_i^u = \sum_{j=1}^n S_{i,j}^u$. We rewrite the vector $\mathbf{r}_k$ as $\mathbf{r}_k = (\mathbf{r}_k^l, \mathbf{r}_k^u)$ where vector $\mathbf{r}_k^l$ refers to the ratings that are already provided by the labeled users, and $\mathbf{r}_k^u$ refers to the ratings to be predicted for the unlabeled users. Similarly, we rewrite L^u and S^u as

$$L^u = \begin{pmatrix} L_{l,l}^u & L_{l,u}^u \\ L_{u,l}^u & L_{u,u}^u \end{pmatrix}, \; S^u = \begin{pmatrix} S_{l,l}^u & S_{l,u}^u \\ S_{u,l}^u & S_{u,u}^u \end{pmatrix} \tag{3.16}$$

where the sub-indices l and u stand for the training users and the test users. Using the above notations, we have the optimal solution to Equation (2.4) written as:

$$\mathbf{r}_k^u = \left[L_{u,u}^u\right]^{-1} S_{u,l}^u \mathbf{r}_k^l. \tag{3.17}$$

The key advantage of using the label propagation method for collaborative filtering lies in its capability of utilizing the similarity information of *all* the training users, including both the rated users and the unrated users, to predict the rating for the test user. In this aspect, the label propagation method for collaboration information is similar to the clustering approach and the collaborative filtering methods based on the eigenvector analysis [40] and the matrix factorization [44, 36].

The problem with the label propagation approach for collaborative filtering is that it is unable to exploit the matrix S^o, the similarity information of items. To address this problem, we propose the relation propagation framework that effectively exploits the similarity of users and the similarity of items, simultaneously. To this end, we modify the energy

function in Equation (2.4) into as follows:

$$E^r = \sum_{i,j=1}^{n} \sum_{k,l=1}^{m} S^u_{i,j} S^o_{k,l} \left(R_{i,k} - R_{j,l}\right)^2 \tag{3.18}$$

In the above energy function, we weight the squared difference $(R_{i,k} - R_{j,l})^2$ by the product between the user similarity $S^u_{i,j}$ and the item similarity $S^o_{k,l}$. In other words, the rating $R_{i,k}$ and $R_{j,l}$ should be close if user u_i share similar interest as user u_j (i.e., $S^u_{i,j}$ is large), and item o_k is similar to item o_j in their descriptive features. It is important to note that the energy function in Equation (3.18) returns back to the energy function in Equation (2.4) when the item similarity matrix S^o becomes an identity matrix, i.e., $S^o_{k,l} = \delta(k,l)$. This is because

$$\begin{aligned} E^r &= \sum_{i,j=1}^{n} \sum_{k,l=1}^{m} S^u_{i,j} \delta(k,l) \left(R_{i,k} - R_{j,l}\right)^2 \\ &= \sum_{k=1}^{m} \sum_{i,j=1}^{n} S^u_{i,j} \left(R_{i,k} - R_{j,k}\right)^2 \\ &= \sum_{k=1}^{m} E^l_k \end{aligned}$$

To minimize the energy function in Equation (3.18), we first introduce a matrix $S^{u,o} = [S^{u,o}_{(i,k),(j,l)}]_{mn \times mn}$ that represents the similarity between two ratings. Each element $S^{u,o}_{(i,k),(j,l)}$ represents the similarity between the rating of item o_k by user u_i and the rating of item o_k by user u_j. According to Equation (3.18), the similarity $S^{u,o}_{(i,k),(j,l)}$ should be defined as $S^u_{i,j} S^o_{k,l}$. In the matrix form, this is equivalent to defining matrix $S^{u,o}$ as the direct product of S^u and S^o, i.e.,

$$S^{u,o} = S^u \bigotimes S^o$$

where operator $\bigotimes$ is the direct product of matrices. Then, the energy function in (3.18) can

be written as

$$E^r = \mathbf{r}^\top L^{u,o} \mathbf{r} \tag{3.19}$$

where

$$\begin{aligned} L^{u,o} &= D^{u,o} - S^{u,o} \\ \mathbf{r} &= (\tilde{\mathbf{r}}_1^\top, \tilde{\mathbf{r}}_2^\top, \ldots, \tilde{\mathbf{r}}_n^\top)^\top \end{aligned}$$

$D^{u,o}$ in the above expression is defined as a diagonal matrix whose diagonal element is defined as

$$\begin{aligned} D^{u,o}_{(i,j),(i,j)} &= \sum_{k=1}^{n} \sum_{l=1}^{m} S^{u,o}_{(i,j),(k,l)} \\ &= \left(\sum_{k=1}^{n} S^u_{i,k} \right) \left(\sum_{l=1}^{m} S^o_{j,l} \right) = D^u_i D^o_j \end{aligned} \tag{3.20}$$

Similar as before, we have the rating vector $\mathbf{r}$ rewritten as $\mathbf{r} = (\mathbf{r}_l, \mathbf{r}_u)$ where $\mathbf{r}_l$ consists of the ratings provided by the training users, and $\mathbf{r}_u$ represents the ratings to be predicted for the test users. Similar to Equation (3.16), we can decompose $S^{u,o}$ and $L^{u,o}$ into the parts related to the training users and the parts related to the test users. Finally, similar to the solution in Equation (3.17), the optimal solution for $\mathbf{r}_u$ that minimizes the energy function in Equation (3.19) is

$$\mathbf{r}_u = \left[L^{u,o}_{u,u} \right]^{-1} S^{u,o}_{u,l} \mathbf{r}_l \tag{3.21}$$

Furthermore, since the normalized graph Laplacian usually delivers better performance than the unnormalized ones, we replace the similarity matrix $S^{u,o}$ and the graph Laplacian

$L^{u,o}$ in the above with the normalized ones, i.e., $\bar{S}^{u,o}$ and $\bar{L}^{u,o}$, that are defined as follows

$$\bar{S}^{u,o} = (D^{u,o})^{-1/2} S^{u,o} (D^{u,o})^{-1/2} \tag{3.22}$$

$$\bar{L}^{u,o} = (D^{u,o})^{-1/2} L^{u,o} (D^{u,o})^{-1/2} = I - \bar{S}^{u,o} \tag{3.23}$$

The solution to Equation (3.18) that uses the normalized similarity matrix is

$$\mathbf{r}_u = \left[\bar{L}^{u,o}_{u,u}\right]^{-1} \bar{S}^{u,o}_{u,l} \mathbf{r}_l \tag{3.24}$$

$L^{u,o}_{u,u}$ in the above can also be written as:

$$\bar{L}^{u,o}_{u,u} = I - \bar{S}^{u,o}_{u,u} = I - \bar{S}^{u}_{u,u} \bigotimes \bar{S}^{o}$$

where $\bar{S}^{u}_{u,u}$ and $\bar{S}^{o}$ are the normalized similarity matrix for training users and the normalized similarity for items, respectively. They are defined similarly as the ones in Equation (3.22).

Remark: The solution in Equation(3.24) can be interpreted from the viewpoint of graph. This graph is constructed by having each node $v_{(i,j)} = (u_i, o_j)$ in the graph corresponding to the rating of item o_j by user u_i. Any two nodes $v_{(i,k)}$ and $v_{(j,l)}$ in the graph are connected by an edge whose weight depends on both the similarity between users u_i and u_j, and the similarity between items o_k and o_l. Then, the above solution can be interpreted as propagated the rating information from the rated items to the unrated ones. It is important to note that the proposed relation propagation framework distinguishes the label propagation approach in that it allows the rating information to propagate among different users and different items simultaneously.

Similar to the multi-label learning model described before, directly computing the solution in Equation (3.24) can be computationally expensive because size of matrix $L^{u,o}_{u,u}$ is $O(mn \times mn)$. It is very hard to compute the matrix inverse given a large number of users

and items. We already present an efficient algorithm for multi-label learning model. In this section, we will solve the problem in a different way which actually leads to the same solution as before.

To resolve this problem, first notice that the solution in (3.24) in fact corresponds to the optimal solution to the following optimization problem:

$$l = -\frac{1}{2}\mathbf{r}_u^\top \left[\bar{L}_{u,u}^{u,o}\right] \mathbf{r}_u + \mathbf{r}_u^\top \bar{S}_{u,l}^{u,o} \mathbf{r}_l \tag{3.25}$$

This can be easily verified by setting the first order derivative of the objective function l to be zero. Thus computing $\mathbf{y}_u$ becomes the problem of optimizing l. By the above conversion, we avoid the difficulty in computing the inverse of $\bar{L}_{u,u}^{u,o}$. However, we still have to resolve the problem with storing the matrix $\bar{L}_{u,u}^{u,o}$ since it is too large. To this end, we assume that the solution $\mathbf{r}_u$ in (3.24) can be written as a linear combination of the direct products between the principle eigenvectors of $\bar{S}_{u,u}^{u}$ and the principle eigenvectors of $\bar{S}^o$.

More specifically, let the top eigenvalues and eigenvectors of $\bar{S}_{u,u}^{u}$ denoted by

$$\left(\lambda_i^u, \mathbf{v}_i^u\right), i = 1, 2, \ldots, K_u$$

and the top eigenvalues and eigenvectos of $\bar{S}^o$ denoted by

$$\left(\lambda_j^o, \mathbf{v}_j^o\right), j = 1, 2, \ldots, K_o$$

where K_u and K_o are the number of eigenvectors extracted from $\bar{S}_{u,u}^{u}$ and $\bar{S}^o$. Then, following our assumption, we can write $\mathbf{r}_u$ as

$$\mathbf{r}_u = \sum_{i=1}^{K_u} \sum_{j=1}^{K_o} \alpha_{i,j} \mathbf{v}_i^u \bigotimes \mathbf{v}_j^o \tag{3.26}$$

where $\alpha_{i,j}$s are the parameters needed to be determined. Substituting the above expression

for $\mathbf{r}_u$ in (3.25), we have the objective function l rewritten as follows:

$$\begin{aligned} l = & -\frac{1}{2}\sum_{i,k=1}^{K_u}\sum_{j,l=1}^{K_o}\alpha_{i,k}\alpha_{j,l}\left[\left(\mathbf{v}_i^u \bigotimes \mathbf{v}_j^o\right)^\top \left(L_{u,u}^{u,o}\right)\left(\mathbf{v}_k^u \bigotimes \mathbf{v}_l^o\right)\right] \\ & + \sum_{i=1}^{K_u}\sum_{j=1}^{K_o}\alpha_{i,j}\left(\mathbf{v}_i^u \bigotimes \mathbf{v}_j^o\right)^\top \left(S_{u,l}^{u,o}\mathbf{r}_l\right) \end{aligned}$$

Using the property of direct product, i.e.,

$$(A \otimes B)(C \oslash D) = (AC \otimes BD),$$

and the relation $\bar{L}_{u,u}^{u,o} = I - \bar{S}_{u,u}^{u} \otimes S^o$ and $\bar{S}_{u,l}^{u,o} = \bar{S}_{u,u}^{u} \otimes \bar{S}^o$, we have the following simplification:

$$\begin{aligned} \left(\mathbf{v}_i^u \bigotimes \mathbf{v}_j^o\right)^\top \left(L_{u,u}^{u,o}\right)\left(\mathbf{v}_k^u \bigotimes \mathbf{v}_l^o\right) &= (1-\lambda_i^u\lambda_j^o)\delta(i,k)\delta(j,l) \\ \left(\mathbf{v}_i^u \bigotimes \mathbf{v}_j^o\right)^\top \left(\bar{S}_{u,l}^{u,o}\mathbf{r}_l\right) &= \lambda_j^o\left(\left[\mathbf{v}_i^u\right]^\top \bar{S}_{u,l}^{u}\right) \bigotimes \left[\mathbf{v}_j^o\right]^\top \mathbf{r}_l \end{aligned}$$

Using the above simplification, the objective function l in the above can be expanded as follows:

$$\begin{aligned} l &= -\frac{1}{2}\sum_{i=1}^{K_u}\sum_{j=1}^{K_o}\alpha_{i,j}^2(1-\lambda_i^u\lambda_j^o) \\ &+ \sum_{i=1}^{K_u}\sum_{j=1}^{K_o}\alpha_{i,j}\lambda_j^o\left(\left[\mathbf{v}_i^u\right]^\top \bar{S}_{u,l}^{u}\right) \bigotimes \left[\mathbf{v}_j^o\right]^\top \mathbf{r}_l \end{aligned} \tag{3.27}$$

Then, the optimal solution for $\alpha_{i,j}$ that maximizes the above objective function is

$$\alpha_{i,j} = \frac{\lambda_j^o}{1-\lambda_i^u\lambda_j^o}\left(\left[\mathbf{v}_i^u\right]^\top \bar{S}_{u,l}^{u}\right) \bigotimes \left[\mathbf{v}_j^o\right]^\top \mathbf{r}_l \tag{3.28}$$

Note that if we substitute $\alpha_{i,j}$ to the equation (3.26), we get the same solution as before.

3.3.2 Case Study I: Movie Recommendations

The goal of our experiment is to evaluate the effectiveness of the Relation Propagation model given the sparse data. We compare the relation propagation model with five baselines: Pearson Correlation Coefficient algorithm (PCC), Personality Diagnosis model (PD), Aspect model (Aspect), Flexible Mixture model (FMM) and Label Propagation model (LP) as described in the previous section.

Dataset

We use the dataset of movie ratings, named MovieLen(http://www.grouplens.org/), as our test bed. MovieLen data set includes 943 users and 1682 movies. Each movie is rated between 1 and 5, with 5 as the best rating. The average number of ratings provided by each user is around 374. To demonstrate the problem of sparse data, we selected 200 training users and for each test user, only 5,10 or 20 movies are randomly selected as the rated items.

Evaluation Metrics

Since one of the goals of collaborative filtering is to rank the unseen movies for a test user, we will apply each of the five algorithms in comparison with order the unseen movies based on their predicted preference, and evaluate these algorithms based on their ranking lists. The evaluation metrics used in our experiments is the average precision [16] (AP), which is a commonly used metric in the information retrieval. It measures the agreement between the top K movies that are ranked by the automatic algorithms, and the top K movies that are ranked by the test user. To this end, we will first rank movies in a descending order of their true ratings by a test user, and then compute the precision P_K of an algorithm for the test user as follows:

$$P_K = \frac{1}{K} \sum_{k=1}^{K} \frac{k}{rank(t_k)} \tag{3.29}$$

where t_k is the k-th movie ranked by the true ratings of the test user. Since the users' ratings only provide a partial order of movies, we break the tie by choosing a random ordering among the movies with the same rating. Finally, the average precision is computed by taking the average of the precision at K over all test users. Evidently, the higher the average precision, the better the performance is. Notice that we did not employ the Mean Average Error (MAE) for evaluation as a number of collaborative filtering papers did. This is because our algorithm is only able to generate the ranking list of items for individual test users. It is unable to predict the rating categories for items, and as a result, we can not measure the MAE metric.

Finally, each experiment is conducted for 10 times and the results across 10 trials are used as the final evaluation metric.

Parameter Setup

Our algorithm needs the inputs S^u, S^o, K_u and K_o. The user similarity matrix S^u is computed using the RBF kernel function $f(r_i, r_j) = exp(-\|r_i - r_j\| / \bar{d})^2$ based on the user rating vectors r_i and r_j, where $\bar{d}$ is the average distance between any two rating vectors and λ is the scaling factor. The movie similarity matrix S^o is obtained by representing each movie as a binary vector in the space of movie categories and using the cosine similarity between the movie vectors as the similarity measurement. Due to the space limitation, we are not going to discuss the impact on the proposed algorithm from different values of the parameters. To speed up our computation, we set the number of eigenvectors $K_u = K_o = 20$ for all experiments according to the empirical experience. We measure the average precision over the top 20 ranks for all algorithms.

Experiment Results

Table 3.1 shows the average precision for the top 20 ranked movies given 5, 10 and 20 observed ratings provided by each test user. First, we can see that the average precision

for all approaches increases as the number of observed movie increases. This is expected since more training data should lead to better performance. Second, the label propagation approach improves the ranking accuracy when compared to the Pearson correlation coefficient approach. However, the label propagation approach did not perform as well as the Aspect Model and the Flexible Mixture Model. We believe this is related to the capability of the algorithms in exploring the correlation among users and the correlation among movies. Both the Aspect Model and the Flexible Mixture Model explore the movie correlation and user correlation by data clustering. In contrast, the label propagation approach only explores the correlation among movies. Third, the relation propagation (RP) algorithm outperforms the other approaches considerably. Most noticeably, the ranking precision of the relation propagation algorithm is 56.7% when the number of rated movies for the test user is 5. This number is better than the ranking precision of the other comparative algorithms when 20 movies are rated by test users. We attribute the success of the relation propagation algorithm to the fact that the proposed algorithm allows relevance information to be propagated among users and movies simultaneously. This is particularly critical to alleviate the problem of sparse data. Although both the Flexible Mixture Model and the Aspect model are able to alleviate the sparse data problem by user clustering, they are limited in that all the users in the same class have the same rating for the same movies. The proposed algorithm relieve this limitation by the propagation scheme. We also verified the average precision across 10 fold is statistically significant at the level of $p < 0.05$ based on the student's t test.

Furthermore, we vary the number of top ranked movies that are rated by the users. Figure 3.6 shows the average precision for different number of top ranked movies for each algorithm given 5, 10 and 20 observed movies for each test user. Again, we observe that for all cases, the proposed relation propagation algorithm is able to outperform the other algorithm considerably. This result again shows the power of the proposed algorithm.

Alg.	Given 5	Given 10	Given 20
RP	**0.567**±(0.003)	**0.569**±(0.007)	**0.571**± (0.020)
LP	0.434±(0.004)	0.446±(0.003)	0.462±(0.004)
FMM	0.489±(0.003)	0.497±(0.001)	0.517±(0.004)
PD	0.470±(0.001)	0.485±(0.001)	0.528±(0.001)
PCC	0.321±(0.002)	0.358±(0.001)	0.433±(0.004)
ASPECT	0.449±(0.002)	0.465±(0.007)	0.515±(0.001)

Table 3.1: Average Precision for the top 20 ranked movies for all five algorithms. Variance of the average precision is included inside the parentheses (all values need to be multiplied with 10^{-2}).

3.3.3 Case Study II: Book Crossing

Book-crossing dataset was collectedc by Cai-Nicolas Ziegler in a 4-week crawl from the Book-crossing community. It contains 278858 users providing 1149780 ratings about 271379 books. Ratings are either explicit, expressed by the scale 1 - 10 (higher values show the higher appreciation), or implicit, expressed by zero.

We sort the books by the number of ratings given to them and select the top 1000 books with the most ratings. Based on these books, we remove users with less than 30 ratings. The resulting dataset has 160 users with 7612 ratings on 1000 books. The average number of ratings for each user is around 47 and the average number of ratings for each book is around 8. In each experiment, 20% of the users are randomly selected as training users and the rest are for testing. For each test user, 5, 10 and 20 books ratings are randomly selected as observed ratings. Each experiment was conducted 10 times and the average across 10 fold was taken as the final value. Average precision is also used to evaluate the performance.

Experiment Setup

Similar to the movie recommendation case, we compare our method to other four algorithms: Pearson Correlation Coefficient (PCC), Aspect model (Aspect), Label Propagation (LP) and Personality Diagnosis model (PD). To be consistent with the evaluation in movie recommendation case, we also use *Average Precision* described in Equation (3.29) as the

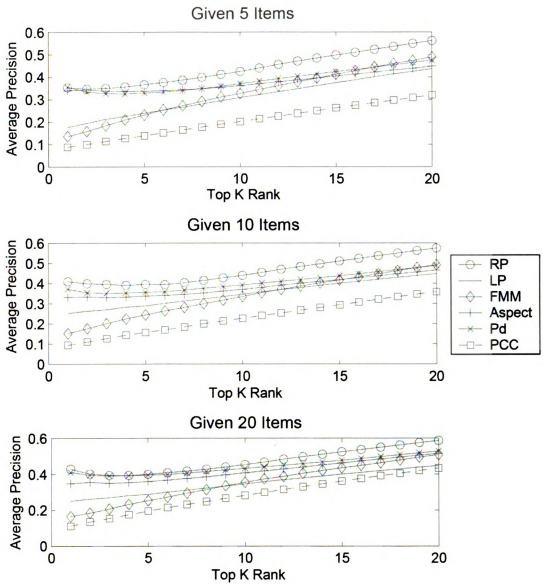


Figure 3.6: Average Precision for different number of top ranked items by the test user given 5,10 or 20 movies.

evaluation metrics. We first compute the average precision on top 10 ranked books for each user and then take the average across all users as the final results.

The relation propagation model requires the input, the user similarity matrix S^u, the book similarity matrix S^o, and the number of eigenvectors K_u and K_o. S^u is computed by taking the dot product between two user rating vectors as $S^u_{ij} = \mathbf{r}_i \cdot \mathbf{r}_j$. Similarly, S^o is computed by first presenting each book as a vector of ratings from all users and then taking

Alg.	Given 5	Given 10	Given 20
RP	0.526±(0.003)	0.569±(0.115)	0.718±(0.051)
LP	0.354±(0.125)	0.426±(0.106)	0.625±(0.334)
PD	10.516±(0.007)	0.556±(0.023)	0.705±(0.001)
PCC	0.403±(0.182)	0.448±(0.138)	0.623±(0.099)
ASPECT	0.480±(0.220)	0.524±(0.201)	0.686±(0.139)

Table 3.2: Average Precision for the Top 10 Ranked Books for all Five Algorithms. Note: Variance of the average precision is included inside the parentheses (all values need to be multiplied with 10^{-3}).

the dot product between two book rating vectors $\widetilde{\mathbf{r}}_i$ and $\widetilde{\mathbf{r}}_j$ as $S^o_{ij} = \widetilde{\mathbf{r}}_i \cdot \widetilde{\mathbf{r}}_j$. According to the empirical experience, we set $K_u = K_o = 20$.

Experiment Results and Analysis

Table 3.2 shows the average precision on the top 10 ranked books for all the algorithms. First, as the number of observed books increases, the average precision increases for all five algorithms. This is expected since more observed ratings provide more information for predicting the test data. Second, LP algorithm performs the worst among all algorithms. This shows that propagating the ratings by using only user similarity may not be enough for accurate prediction. Simple PCC method works better than LP algorithm general although worse than other approaches. Third, PD model and Aspect model work better than LP model and PCC method, but worse than the RP model. Especially the performance of the PD model is very close to the RP model. Grouping users by the similarity of their personality seems a effective strategy to improve the performance. Finally, the RP model performs the best among all the approaches. We believe it's the contribution of exploring the correlation among users and books. All results are statistically significant at 95% level. Figure 3.7 shows the average precision at different number of top ranked books given 5, 10 and 20 observed ratings. The RP model performs best at each different rank and the result is consistent with the above analysis.

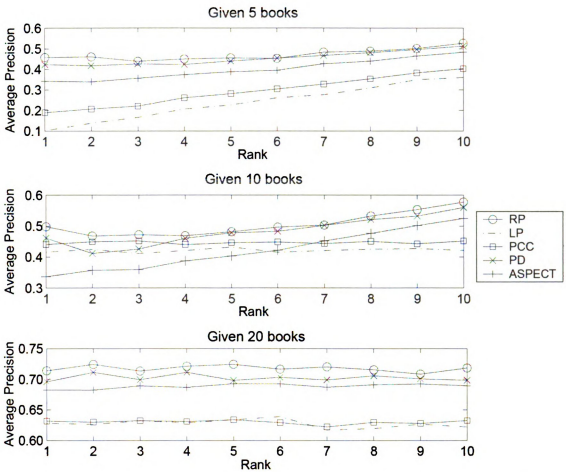


Figure 3.7: Average Precision for Different Number of Top Ranked Books by the Test User Given 5,10 or 20 Books.

3.4 Drawbacks of Label Propagation Approaches

Label propagation approach has attracted a lot of attention from researchers in the machine learning area with its promising performance in many applications including information retrieval, classification and ranking problems. In the previous sections, motivated by the label propagation approach, we presented the relation propagation model and their applications. Empirical studies showed that it is very helpful in a lot of problems to propagate the class (or the rating) information on the weighted graph which is appropriately constructed. However, there are still some general issues with the traditional label propagation approaches.

The label propagation approach essentially addresses a ranking problem. In order to apply the label propagation approach, we construct the graph based on the given data including training and testing examples. Then the labeling information on the training data is propagated through the graph such that the entire system reaches a global equilibrium. Each node in the graph will get a corresponding numeric confidence score computed through the propagation. Finally we compute a ranking list based on the confidence score assigned to each node and make the prediction accordingly. Although it sounds a practical idea, the problem comes from propagating ordinal values (or class labels) as numerical values.

In many cases, the labeling information for data examples does not carry the numerical meaning. The label values are usually discrete and finite. For instance, in a classification problem, each class label is represented by a number that should carry only ordinal meaning. Consider the case that there are three classes $c_1 = 1$, $c_2 = 2$ and $c_3 = 3$. We can not say that, the class c_1 should be counted less than c_2 and c_3 because numerically c_1 is smaller than c_1 and c_2. Furthermore there is not order among c_1, c_2 and c_3 and changing the label value for each class will not change the nature of the problem. The class numbers are only the names for the classes. But during the process of propagation, we actually propagate the labeling information that is supposed to be ordinal values as numerical value and take the final numerical results as the base of assigning ordinal labels. This is also the case in a lot

of ranking problem. For example, in collaborative filtering task, each rating is often represented by a number. Different from class labels, rating information does exhibit an order among different categories. If we have three categories $bad(1)$, $good(2)$ and $excellent(3)$, we can tell the rating $1 < 2 < 3$. However, propagating the rating information as numerical values is not an appropriate practice since those rating numbers only carry the ordinal meaning. Replacing the rating number with any other numerical value without breaking the ordering will not change the nature of the problem, but it may significantly affect the final results.

In a lot of ranking problems, it's also very common that the exact rating information is not available. Instead, only the ranking information is given for the observed data. For instance, a web search engine presents a ranking list to the user and the user may click the preferred web pages. This clicked-through history for the user generates the partial preference judgements over the clicked pages and the unclicked pages. This kind of ranked data is a challenge for traditional classification and regression approaches. Without the exact ratings, propagating the rating information is not practical at all.

In a word, label propagation has shown itself a promising approach to many problems while the issues we discussed above may limit its efficiency in a lot of scenarios. In the next chapter, we will discuss a solution which addresses the challenge of the ranked data for label propagation.

Chapter 4

Propagation for Ranked Data

Analysis of ranked data is often involved in practical applications including information retrieval, collaborative filtering, etc. For example, the ranking presented by a search engine to the user is a ranking list of all web pages based on the relevance of each page to the user's query; the links clicked by a user provide some kind of relevance judgements for web pages; the user's past ratings on movies are also ranked data in terms of the user's preferences. The interest in ranking problems is still the source of ongoing research in many fields such as mathematical economics, social science and computer science.

Ranking problems differ from traditional classification and regression problems in that ranking problems explore the ordering of examples rather than the absolute numeric/ordinal values assigned to each example. There are two different views of ranking problems. In the first view, the training examples are presented with the ranking scores (e.g., relevance judgements or rank scales) and the goal is to learn a score function from the labeled data examples. One common approach for this kind of ranking problem is to learn a probabilistic classifier or regression model from the given ranking scores. However, This ranking problem may be hard for regression models since the given ranks are usually discrete and finite. It is also unsuitable for classification models since the ranking information does not exist in class labels. There are a number of works devoted to this area [43, 18, 42, 12].

In [18], the authors proposed a large margin algorithm based on a mapping from objects to scalar utility values using a distribution independent risk formulation of rank learning. Kramer et al. investigated the use of a regression tree learner by mapping the ordinal variables into numerical values [38]. In [15], the authors converted a rank learning problem into nested binary classification problems that encode the ordering of the original ranks and then the results of standard binary classifiers can be organized for prediction. In [43], two large margin principle-based approaches were proposed for ranking learning in which the rankings are viewed as ordinal scales. The first appraoch uses the "fixed margin" policy in which the margin of the closest neighboring classes is being maximized and the second approach allows for multiple different margins where the sum of margins is maximized. [9] presented a probabilistic kernel approach to ranking learning based on Gaussian processes.

In the second view, the training data is presented in the format of partial ordering information (e.g., preference judgements) and the goal is to learn a ranking function from the training data examples which generates the ordering of the data. The partial ordering information can be provided in the form of the ranking list of training instances or the preferences over pairs on training set. Given the ranking of training instances, the ranking function can be learned by minimizing the difference between the given ranking and predicted ranking [22, 32]. In [32], an approach was introduced to take ranking rather than classification as fundamental. This approach uses a generalization of the Mallows model on permutations to combine multiple input rankings. In [22], the author presented a Support Vector Machine approach which automatically optimizes the retrieval quality of search engines using clickthrough data. The ordering information can also be broken into a set of pairwise preference judgements and used in the form of pairwise preferences [25]. Given the pairwise preference judgements, we can construct the models which predict the preferences or the rankings [16, 11, 31]. These kinds of approaches appear more appropriate in ranking problems since we focus on the relative ordering information rather than absolute values. An advantage of learning ordering directly is that preference judgements

can be much easier to obtain than the exact labels required for classification or numeric values for regression. This chapter focuses on the approaches utilizing the relative ordering information.

Although there has been a significant amount of research done on ranking models in statistics and machine learning, there is little work on label propagation models for ranked data. However, it is very important and also a natural decision to explore the label propagation approaches for ranked data due to the existing challenges discussed in section 3.4. One of the challenges is related to the meaning carried by the class labels. As we described above, the label propagation approaches address all the problems by first propagating the class labels and then generating ranking lists based on the resulting propagation scores computed by the algorithm. This may not be an appropriate approach in certain cases since the numerical values are not able to preserve the essential meaning of the ordinal information among class labels. For example, consider the numerical class labels 1, 2, 3, $\cdots$. In most cases, the numbers just provide the labeling for classes instead of the ordering information. Class 1 is not greater or smaller than class 2. However, when the class labels are propagated, the ordering information will be introduced into the propagation process. Another challenge is the difficulty of obtaining the exact labels of the labeled data. Most previous studies require the *absolute* label values or numeric values to be given in order to learn the ranking function, which is often not practical. The absolute labeling information is usually hard to obtain or is very costly and time consuming. In contrast, the partial *relative* ordering information is more easily available for training in some cases. For instance, in a movie rating dataset, the users provide the ratings which indicate the preferences of certain movies over other movies. Traditional label propagation approaches may not fit in these applications because they are required to propagate the absolute labels. In order to address these challenges, this chapter is devoted to label propagation on ranked data.

The rest of the chapter will start with the formal definition of the ranking problem on relative ordering information, followed by the discussion of some recent research works

related to rank learning. Then it will propose a supervised framework of *Rank Propagation* in which the model is learned from pairwise preference judgements.

4.1 Ranking Problem Setting

Let's formally describe the ranking problem over the relative ordering information. Let $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ be a set of labeled and unlabeled instances called the *domain* or *instance space*. Elements of $\mathcal{X}$ are called *instances*. Each instance $x_i \in \mathbf{R}^d$ $(i = 1, \cdots, n)$ is represented by a vector in d dimensional space.

We define *preference judgement function* Φ which is the input to the learning algorithm. This function encodes the known relative ordering information about a subset (training set) of the domain $\mathcal{X}$. For instance, in the movie recommendation task, the feedback from a movie reviewer provides the known movie preferences by this reviewer. Formally, we define the preference judgement function which has the form $\Phi : \mathcal{X} \times \mathcal{X} \rightarrow \mathbf{R}$. $\Phi(x_0, x_1)$ represents the degree to which x_1 should be correctly ranked above x_0. The positive value means that x_1 should be ranked higher above x_0 and negative value means that x_1 should be ranked below x_0. We also define $\Phi(x, x) = 0$ and Φ is anti-symmetric in the sense that $\Phi(x_0, x_1) = -\Phi(x_1, x_0)$ for all $x_0, x_1 \in \mathcal{X}$.

The learning algorithm will output a ranking of all instances represented in the form of a function $H : \mathcal{X} \rightarrow \mathbf{R}$. We interpret $H(x_0) > H(x_1)$ as x_0 is ranked above x_1 and vice versa.

The goal of the learning algorithm is to produce a ranking function H which generates a good ranking based on the $perferencejudgements$ encoded by Φ.

4.2 Related Works

There are a number of studies devoted to the analysis of ranked data. This section will introduce some literature in the learning approaches on ranked data in machine learning

area.

4.2.1 Statistical Measure of Rank Correlation

Guttman's Point Alienation, a statistical measure of rank correlation, has been used to construct the objective function that matches user's preferences [2] in a combination expert model. Previous work [1] has demonstrated that this measure can be highly correlated with average precision, which is a more typical measure of performance in information retrieval. Thus optimization of this objective function is likely to lead to the optimized average precision performance. The objective function can be optimized by gradient descent even though there are singularities.

Let $R_{\Theta,q}(d)$ be a ranked retrieval function which generates a score indicating the relevance of document d to a query q (R must be differentiable in its parameters Θ). The objective function is:

$$J(R_\Theta) = \frac{-1}{Q} \sum_{q \in Q} \frac{\sum_{d >_q d'} (R_{\Theta,q}(d)) - R_{\Theta,q}(d')}{\sum_{d >_q d'} |R_{\Theta,q}(d) - R_{\Theta,q}(d')|} \tag{4.1}$$

where Q is the set of training queries and d and d' are documents retrieved. $>_q$ is defined as a preference relation over document pairs as

$$d >_q d' \iff \text{the user prefers } d \text{ to } d'$$

The goal is to find parameters Θ so that $R_{\Theta,q}(d) > R_{\Theta,q}(d')$ whenever $d >_q d'$. This method is applied to combining experts as the following linear model (three experts are illustrated):

$$R_{\Theta,q}(d) = \Theta_1 E_1(q,d) + \Theta_2 E2(q,d) + \Theta_3 E_3(q,d)$$

The parameters Θ_i serves as the scales on individual expert which is denoted as E_i. The critical feature of such a model is that the score it generates is differentiable with respect to

the parameters Θ. The gradient in general is:

$$\frac{\partial J(R_\Theta)}{\partial \Theta} = \sum_{q \in Q} \sum_{d \in D_q} \frac{J(R_\Theta)}{\partial R_{\Theta,q}(d)} \cdot \frac{\partial R_{\Theta,q}(d)}{\partial \Theta}$$

In the experiment on a commercial retrieval system called CMME, the combination model performs 47% better than any single expert.

4.2.2 Ranking SVM

Support Vector Machine approach has been applied to ranking problem for automatically optimizing the retrieval quality of search engines using clickthrough data in [22]. This method is shown to be well founded in a risk minimization framework.

Clickthrough data in search engine can be viewed as triplets (q, r, c) where q is the query, r is the ranking presented by the search engine and c is the set of links the user clicked on. Since the clicked-on link set c depends on the query q and the presented ranking r, partial relative relevance judgement would be a better interpretation for clickthrough data than absolution relevance. The problem is formalized in this approach as: Given a query q and a document set $D = \{d_1, \cdots, d_m\}$, the retrieval system should return a rankings of documents in D according to their relevance with respect to q. Since retrieval systems usually can not achieve an optimal ranking, the retrieval function f are evaluated by how closely its ordering $r_{f(q)}$ approximates the optimum r^*. Both r^* and $r_{f(q)}$ are considered as weak ordering. Kendall's τ [26] is used as performance measure to evaluate the closeness between r^* and $r_{f(q)}$. It is defined as

$$\tau(r_a, r_b) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\frac{m}{2}}$$

where P is the number of concordant pairs and Q is the number of discordant pairs between r_a and r_b. m is the number of documents on a finite domain D and the sum of P and Q is $\frac{m}{2}$

for strict ordering. It can be proved that maximizing $\tau(r_{f(q)}, r^*)$ is connected to improved retrieval quality. Given an independently and identically distributed training sample S of size n containing queries q with their target rankings r^*

$$(q_1, r_1^*), (q_2, r_2^*), \cdots, (q_n, r_n^*).$$

the learning $\mathcal{L}$ will find a ranking function f from a family of ranking functions F that maximizes the empirical τ defined as

$$\tau_S(f) = \frac{1}{n} \sum_{i=1}^{n} \tau(r_{f(q_i)}, r_i^*). \tag{4.2}$$

This SVM ranking algorithm will find the function f out of a family of ranking functions F that maximizes (4.2) and generalizes well beyond the training data. Given the class of linear ranking functions

$$(d_i, d_j) \in f_{\vec{w}}(q) \Longleftrightarrow \vec{w}\Phi(q, d_i) > \vec{w}\Phi(q, d_j)$$

where $\vec{w}$ is a weight vector which needs to be learned and $\Phi(q, d)$ is a mapping onto feature vector describing the match between query q and document d. The optimization problem is defined as follows

OPTIMIZATION PROBLEM 1. (RANKING SVM)

minimize:

$$V(\vec{w}, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \tag{4.3}$$

subject to:

$$l \forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1}$$

$$\ldots$$

$$\forall (d_i, d_j) \in r_n^* : \overrightarrow{w}\Phi(q_n, d_i) \geq \overrightarrow{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \tag{4.4}$$

$$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \tag{4.5}$$

C is a parameter that allows trading-off margin size against training error. Adapting the Ranking SVM to the case of partial relevance data by replacing r^* with the observed preferences r', the optimization problem can be defined similarly.

4.2.3 Large Margin Principle

Large Margin principle has been applied for ranking problem in [43]. The paper views the problem of ranking k instances as predicting variables of ordinal scale and introduced two approaches for learning ranking functions that applied the large margin principle used in SVM to the ordinal regression while maintaining an optimal problem size linear in the number of training examples. The first approach is the "fixed margin" policy in which the margin of the closest neighboring classes is maximized which is a direct generalization of SVM to ranking learning. The second approach allows for $k-1$ different margins where the sum of margins is maximized.

Let $\mathbf{x}_i^j$ be the set of training example where $j = 1, \cdots, k$ denotes the class number and $i = 1, \cdots, i_j$ is the index within each class. $l = \sum_j i_j$ is the total number of training examples. Similar to 2-class separating hyperplane problem, this method is to look for $k-1$ parallel hyperplanes represented by vector $\mathbf{w} \in R^n$ (n is the dimension of the input vectors) and scalars $b_1 \leq \cdots \leq b_{k-1}$. Thus the hyperplanes are defined as $(\mathbf{w}, b_1), \cdots, (\mathbf{w}, b_{k-1})$ such that the training data are separated by the following decision rule:

$$f(\mathbf{x}) = \min_{r \in \{1, \cdots, k\}} \{r : \mathbf{w} \cdot \mathbf{x} - b_r < 0\}$$

In other words, all input vectors $\mathbf{x}$ are assigned the rank r if $b_{r-1} < \mathbf{w} \cdot \mathbf{x} < b_r$.

In order to control the tradeoff between lowering the empirical risk (error measure on the training et) and lowering the confidence interval, the model should maximize the margin between the boundaries of sets according to the structural risk minimization principle. Two approaches based on large margin principle to different optimization problem.

- *fixed margin*: the margin to be maximized is the margin between the closest neighboring sets. If $\mathbf{w}$ and b_r are scaled such that the distance of the boundary points from the hyperplane $(\mathbf{w}, b_r)$ is 1, the margin between the classes r and $r+1$ becomes $2/|\mathbf{w}|$. Thus the goal is to find the direction $\mathbf{w}$ and the scalars $b1, \cdots, b_{k-1}$ such that $\mathbf{w} \cdot \mathbf{w}$ is minimized. This leads to the following optimization problem:

$$\min_{w, b_j, \epsilon_i^j, \epsilon_i^{*j+1}} \quad \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \sum_i \sum_j \left(\epsilon_i^j + \epsilon_i^{*j+1}\right) \tag{4.6}$$

$$\text{s.t.} \quad \mathbf{w} \cdot \mathbf{x}_i^j - b_j \leq -1 + \epsilon_i^j \tag{4.7}$$

$$\mathbf{w} \cdot \mathbf{x}_i^j - b_j \leq -1 + \epsilon_i^j \tag{4.8}$$

$$\mathbf{w} \cdot \mathbf{x}_i^{j+1} - b_j \leq 1 - \epsilon_i^{*j+1} \tag{4.9}$$

$$\epsilon_i^j \geq 0, \epsilon_i^{*j} \geq 0 \tag{4.10}$$

  where $j = 1, \cdots, k-1$ and $i = 1, \cdots, i_j$, and C is predefined constant.

- *sum-of-margins*: large margin policy can be applied to $k-1$ margins such that the sum of them is maximized. The challenge is that pre-scaling of $\mathbf{w}$ can not be adopted as in *fixed margin* strategy. In order to solve this, the primal functional is represented by $2(k-1)$ parallel hyperplanes instead of $k-1$. The goal is to seek a ranking rule which employs a vector $\mathbf{w}$ and a set of $2(k-)$ thresholds $a_1 \leq b_1 \leq a_2 \leq b_2 \leq \cdots \leq a_{k-1} \leq b_{k-1}$ such that $\mathbf{w} \cdot \mathbf{x}_i^j \leq a_j$ and $\mathbf{w} \cdot \mathbf{x}_i^j \geq b_{j-1}$, $b_0 = -\infty$ and $a_k = \infty$. The margin between two hyperplanes separating class j and $j+1$ is: $(b_j - a_j)/\sqrt{\|\mathbf{w}\|}$.

Hence, the optimization problem can formulated as:

$$\min_{w,a_j,b_j} \sum_{j=1}^{k-1}(a_j - b_j) + C\sum_i\sum_j\left(\epsilon_i^j + \epsilon_i^{*j+1}\right) \tag{4.11}$$

$$\text{s.t.} \quad a_j \leq b_j \tag{4.12}$$

$$b_j \leq a_j + 1, j = 1, \cdots, k-2 \tag{4.13}$$

$$\mathbf{w}\cdot\mathbf{x}_i^j \leq a_j + \epsilon_i^j, b_j - \epsilon_i^{*j+1} \leq \mathbf{w}\cdot\mathbf{x}_i^{j+1} \tag{4.14}$$

$$\mathbf{w}\cdot\mathbf{w} \leq 1, \epsilon_i^j \leq 0, \epsilon_i^{*j+1} \leq 0 \tag{4.15}$$

4.2.4 PRank Algorithm

PRank is an online algorithm for ranking problems motivated by the perceptron algorithm for classification. This method projects each instance into the reals and then operates on rankings by associating each ranking with distinct sub-interval of the reals.

Given a sequence $(\mathbf{x}^1, y^1), \cdots, (\mathbf{x}^t, y^t), \cdots$ of instance-rank pairs, $\mathbf{x}^t$ is preferred over $\mathbf{x}^s$ if $y^t > y^s$ and y^t is an element from finite set $\mathcal{Y} = \{1, 2, \cdots, k\}$ with $>$ as the order relation. The ranking rule $H : R^n \rightarrow \mathcal{Y}$ which is a mapping from the instances to ranks. The family of ranking rules H employs a vector $\mathbf{w} \in R^n$ and a set of k thresholds $b_1 \geq \cdots \geq b_{k-1} \geq b_k = \infty$. The predicted rank is then defined to be the index of the first threshold b_r for which $\mathbf{w}\cdot x < b_r$. The goal of the algorithm is to learn the ranking rule minimizing the ranking-loss which is defined to be the number of thresholds between the true rank and the predicted rank.

To learn the ranking rule online, the true rank y is expanded into $k-1$ variables $y_1, \cdots, y_{k-1}$ and y induces a vector as follows:

$$\{y_1, \cdots, y_{k-1}\} = (\underbrace{+1, \cdots, +1}_{r}, -1, \cdots, -1)$$

where r is the maximum index for which $y_r = +1$ and $r = y - 1$. Thus, the predicted rank

is correct if $y_r(\mathbf{w} \cdot \mathbf{x} - b_r) > 0$ for all r, otherwise the rule is updated as:

- replace b_r with $b_r - y_r$ for which $y_r(\mathbf{w} \cdot \mathbf{x} - b_r) \leq 0$.
- replace $\mathbf{w}$ with $\mathbf{w} + (\sum_{y_r})\mathbf{x}$ for which $y_r(\mathbf{w} \cdot \mathbf{x} - b_r) \leq 0$.

The paper proved that *PRank* maintains a consistent hypothesis in the sense that it preserves the correct order of the thresholds and given an input sequence $(\mathbf{x}^1, y^1), \cdots, (\mathbf{x}^T, y^T)$, the rank loss $\sum_{t=1}^{T} |\hat{y}^t - y^t|$ is at most $(k-1)(R^2+1)/\gamma^2$ where $R^2 = \max \parallel \mathbf{x}^t \parallel^2$ and $\gamma = \min_{r,t}\{(\mathbf{w}^* \cdot \mathbf{x}^t - b_r^*)y_r^t\}$ (see details in [12]).

4.2.5 Conditional Model on Permutations

CRanking [32], a new approach to ensemble learning, explores conditional models on permutations as a tool for solving ranking problems. This approach views each input classifier in terms of the ranked list of labels that it assigns to the input, builds probability distributions over rankings of the labels and then learn the models on permutation groups. The basic model in this approach is an extension of the Mallows model to the conditional setting and it has a natural Bayesian interpretation as a generative model because of the invariance properties of the sufficient statistics.

Let $\mathcal{X} = y_1, \ldots, y_n$ be a set of items to be ranked, identified with the numbers $1, \cdots, n$. A permutation π is a bijection from $\{1, \cdots, n\}$ to itself. Let $\pi(i)$ denote the ranking given to the item i and $\pi^{-1}(i)$ denote the item assigned to rank i. The collection of all permutations of n-items forms a non-abelian group under composition, called the *symmetric group of order n*, and denoted $\mathcal{S}_n$. Given a set of training instances, a ranking of items needs to be assigned to each instance. The permutations given by the j-th input classifier for the i-th instance is denoted as $\sigma_j^{(i)}$. The permutation $\pi^{(i)}$ is used to denote the *predicted* ranking for the i-th instance. σ is used to denote a sequence of permutations σ_j.

A generalization of the Mallows model was proposed for estimating a conditional distribution. Let $\sigma_j \in \mathcal{S}_n$ be a permutation from the j-th input classifier and $\theta_j \in \mathbf{R}$ for

$j = 1, \cdots, k$. The distribution which is described as follows:

$$p(\pi|\sigma, \theta) = \frac{1}{Z(\theta, \sigma)} e^{\sum_{j=1}^{k} \theta_j d(\pi, \sigma_j)}$$

defines a conditional model when there are multiple instances and each instance is associated with a possibly differently set of rankings $\sigma_j^{(i)}$. $Z(\theta, \sigma)$ is the normalizing constant defined as $Z(\theta, \sigma) = e^{\psi(\theta)}$ and ψ is the cumulant function defined as $\psi(\theta, \sigma) = \log \sum_{\pi \in \mathcal{S}_n} \exp(\theta d(\pi, \sigma))$. The model can be applied to, for example, the rankings of web pages generated by different search engines for a specified query. θ_j can be viewed as the weight for each ranker in the ensemble. In this scenario, only the θ_j are the free parameters to be estimated. The likelihood function based on this model is convex in θ. Given a training set consisting of the pairs $D = (\pi^{(i)}, \sigma^{(i)})$, the parameters can be estimated by maximum conditional likelihood or MAP.

This model is different from other discriminative models in that it has a natural Bayesian interpretation. If we view π as a parameter, suppose σ_j are indepently sampled from Mallow models with common mode π and dispersion parameters θ_j. Under a prior $p(\pi)$, the posterior is

$$p(\pi|\theta, \sigma) \propto p(\pi) \exp\left(\sum_j \theta_j d(\pi, \theta)\right)$$

because of the invariance property of $d(.,.)$. Thus, under a uniform prior on π, the distribution is precisely the posterior of a generative model for the rankings σ.

4.2.6 Conditional Model on Ranking Poset

Another conditional model on ranking poset is presented for ranking problem in [31]. Similar to *CRank* algorithm, this model is also an extension of the Mallows ϕ model and it generalizes the classifier combination methods used by several ensemble leanring algorithms, including error correcting output codes, discrete AdaBoost, logistic regression and cranking.

A poset (partially ordered set) is a pair $(P, \leq)$, where P is a set and $\leq$ is a binary relation that satisfies (1)$x \leq x$, (2)if $x \leq y$ and $y \leq x$ then $x = y$, and (3)if $x \leq y$ and $y \leq z$ then $x \leq z$ for all $x, y, z \in P$. Then ranking poset $\mathcal{W}_n$ is defined as the poset in which the elements are all possible cosets $\mathcal{G}_\lambda \pi$, where λ is an ordered partition of n and $\pi \in \mathcal{G}$. The right coset is defined as: $\mathcal{G}_{n-k}\pi = \{\sigma\pi | \sigma \in \mathcal{G}_{n-k}\}$ and $\mathcal{G}_{n-k} = \{\pi \in \mathcal{G}_n | \pi(i) = i, i = 1, \cdots, k\}$ where π denotes a permutation of $\{1, \cdots, n\}$ and $\pi(i)$ denotes the rank given to item i. The right coset can be viewed as a partial ranking, where there is a ful ordering of the k top-ranked items.

Kendall's Tau $T(\pi, \sigma)$ is used as a distance measure d which is define in (4.2). Given input k rankings $\sigma_1, \sigma_2, \cdots, \sigma_k$ contained in some subset $\sigma \in \mathcal{W}_n$ of the ranking poset and a probability mass function q_0 on $\mathcal{W}_n$, an exponential model $p_\theta(\pi|\sigma)$ is given as:

$$p_\theta(\pi|\sigma) = \frac{1}{Z(\sigma, \theta)} q_0(\pi) \text{exp}\left(\sum_{j=1}^{k} \theta_j d(\pi, \sigma_j)\right)$$

where $\theta \in \times \in R_k$, $\pi \in \Pi \in \mathcal{W}_n$ and $\sigma_j \in \Sigma \in \mathcal{W}_n$. The term $Z(\theta, \sigma)$ is the normalizing constant

$$Z(\theta, \sigma) = \sum_{\pi \in \Pi} q_0(\pi) \text{exp}\left(\sum_{j=1}^{k} \theta + j d(\pi, \sigma_j)\right)$$

4.3 Rank Propagation for Multi-label Learning

In this section, we propose the idea of rank propagation for multi-label learning problem. We view the problem of multi-label learning as multi-label ranking, i.e., instead of making a hard decision about which subset of class labels should be assigned to each instance. Our goal is to rank the class labels according to their relevance to the given instance. Unlike the typical label propagation schema that propagate the prediction scores, the key idea of rank propagation is to propagate the pairwise preference relationship of class labels between labeled examples and unlabeled examples. We will focus the discussion on rank

propagation for supervised learning.

4.3.1 Preliminaries

Let $X_l = (\mathbf{x}_1^l, \mathbf{x}_2^l, \cdots, \mathbf{x}_u^l)$ denote the set of labeled examples. Each example $\mathbf{x}_i$ is labeled by a vector $\mathbf{y}_i = (y_{i,1}, y_{i,2}, \cdots, y_{i,C})$ where C is the number of classes and $y_{i,k} \in \{0, 1\}$ indicates if example $\mathbf{x}_i$ is assigned to the kth class (1 indicates the example belongs to the kth class and 0 otherwise). We denote by $Y = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n)$ the class labels assigned to all the instances. Let $X_u = (\mathbf{x}_1^u, \mathbf{x}_2^u, \cdots, \mathbf{x}_m^u)$ denote the set of unlabeled examples. Our goal is to rank the class labels for each unlabeled example.

4.3.2 Preference Matrix

We focus on a single unlabeled example $\mathbf{x}$. We denote by $k(\mathbf{x}, \mathbf{x}')$ the kernel similarity function. For the sake of simplicity, we assume that the kernel similarity function always outputs non-negative values, i.e., $k(\mathbf{x}, \mathbf{x}') \geq 0$ for any $\mathbf{x}$ and $\mathbf{x}'$. To conduct the rank propagation, we encode the class label information into a pairwise preference matrix. More specifically, given assigned class labels $\mathbf{y}_i$, the preference matrix $M_i \in \mathbb{R}^{C \times C}$ is defined as follows:

$$[M_i]_{k,l} = \begin{cases} a & y_{i,l} = 1 \text{ and } y_{i,k} = 1 \\ b & y_{i,l} = 1 \text{ and } y_{i,k} = 0 \\ a & y_{i,l} = 0 \text{ and } y_{i,k} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

where a and b are constants with $a < b$. We can normalize the preference matrix M_i into a transition matrix S_i by defining

$$[S_i]_{k,l} = \frac{[M_i]_{k,l}}{\sum_{k'=1}^{n} [M_i]_{k',l}} \tag{4.17}$$

Evidently, the constant a is proportional to the transition probability between the classes that are either both selected or both unselected. Similarly, constant b is proportional to the transitional probability between the selected classes and the unselected classes. it is not difficult to see that the principle eigenvector of S, i.e., $S\mathbf{v} = \mathbf{v}$,is

$$v_{i,k} = \begin{cases} 1/\|\mathbf{y}_i\|_2 & y_{i,k} = 1 \\ 0 & y_{i,k} = 0 \end{cases}$$

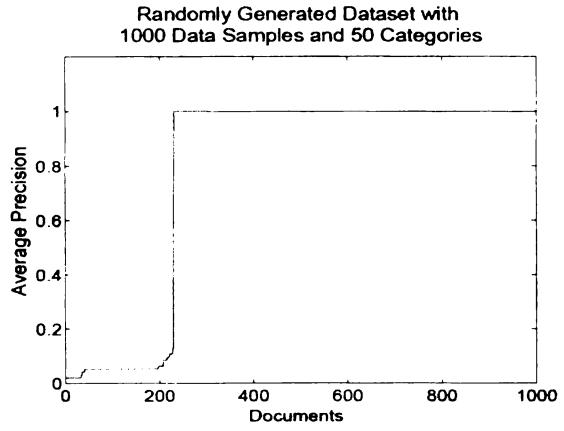
Furthermore, we denote by $S \in \mathbb{R}_+^{C \times C}$ the transitional matrix for the test example $\mathbf{x}$. Our goal is to search for the transitional matrix S that are coherent with the transitional matrix in the neighborhood $\mathbf{x}$.

Principal Eigenvector of Transition Matrix As claimed in the above section, the principle eigenvector of the transitional matrix for each example is consistent with the class assignment of the example. Thus, by computing the principle eigenvector of the transitional matrix of a testing example, we can generate a ranking list of all classes labels for this example and make predictions accordingly.

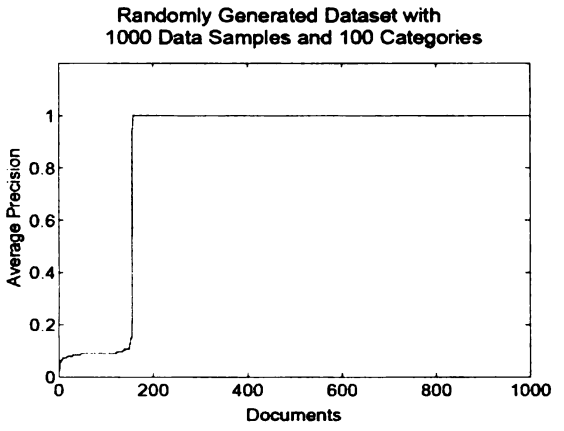
To further verify our assumption, we randomly generated four datasets. Each data sample is assigned to one or multiple categories. We build the preference matrix for each data sample based on the category information as described above and then compute the principal eigenvector. To evaluate how much the ranking of categories based on the principal eigenvector is consistent with the real category information, we use *Average Precision* as the evaluation metric.

Figure 4.2 shows the average precision for each document in all generated datasets. Clearly, for the majority of the data samples, the principal eigenvector of preference matrix is completely consistent with the real category information.

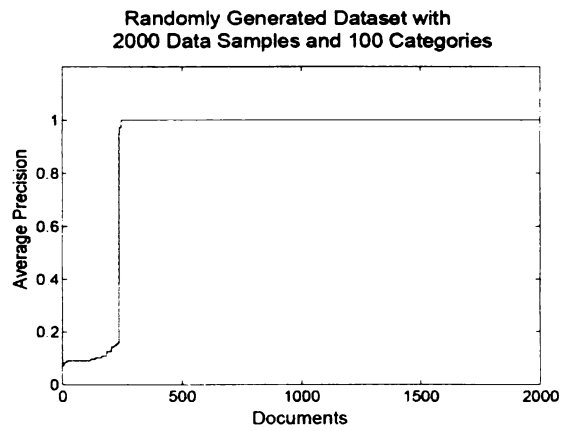
We also verify this assumption with the existing datasets which will be used in our case study discussed in the following section.



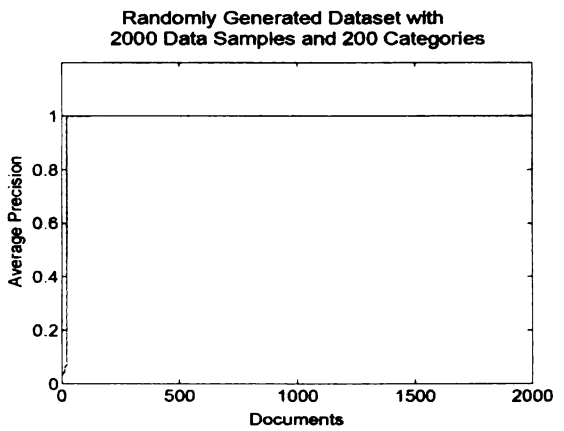
(a) Randomly generated dataset 2: average number of categories per document is 5 with maximum 13 and minimum 0; average number of documents per category is 101 with maximum 127 and minimum 79.



(b) Randomly generated dataset 3: average number of categories per document is 15 with maximum 26 and minimum 4; average number of documents per category is 149 with maximum 181 and minimum 117.

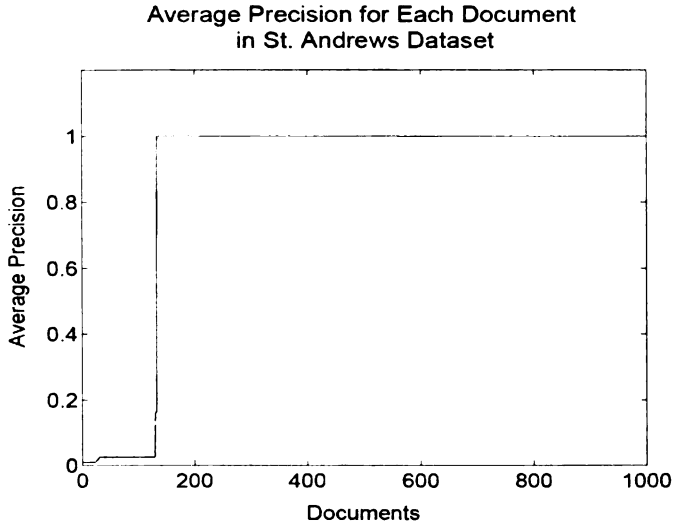


(c) Randomly generated dataset 1: average number of categories per document is 20 with maximum 35 and minimum 9; average number of documents per category is 400 with maximum 448 and minimum 361.

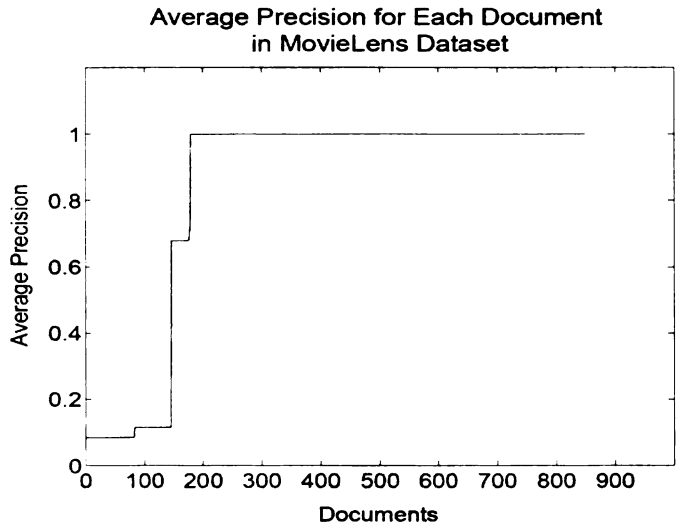


(d) Randomly generated dataset 3: average number of categories per document is 20 with maximum 35 and minimum 8; average number of documents per category is 199 with maximum 239 and minimum 166.

Figure 4.1: Analysis of Principal Eigenvectors of Preference Matrix for Generated Datasets.



(a) *ST. Andrews* dataset: average number of categories per document is 6 with maximum 11 and minimum 1; average number of documents per category is 56 with maximum 740 and minimum 10.



(b) *MovieLens* dataset: average number of categories per document is 3 with maximum 6 and minimum 2; average number of documents per category is 108 with maximum 349 and minimum 1.

Figure 4.2: Analysis of Principal Eigenvectors of Preference Matrix for Study Datasets.

4.3.3 Framework Description

Using the transitional matrix, we view the multi-label learning problem as a multi-label ranking problem. This goal can be encoded into the following optimization problem

$$\begin{aligned} \min_{S} \quad & \sum_{i=1}^{n} l(S, S_i) w_i \\ \text{s. t.} \quad & S_{k,l} \geq 0, \ \forall k, l = 1, 2, \cdots, C \\ & \sum_{k=1}^{C} S_{k,l} = 1 \end{aligned} \tag{4.18}$$

where $w_i = k(\mathbf{x}, \mathbf{x}_i)$. $l(S, S')$ is the loss function that measures the difference between two matrix S and S'. Different types of loss functions can be used in this optimization problem. We consider two choices of loss functions as follows:

l_1 loss function It's defined using the trace function as $l_1(S, S') = \text{tr}((S - S')^\top (S - S'))$. This loss function can be further rewritten as

$$l_1(S, S') = \sum_{k,l=1}^{C} (S_{k,l} - S'_{k,l})^2. \tag{4.19}$$

Obviously, the above equation essentially measures the square of the difference between two matrices across all the elements.

The solution of S is straightforward when using the first loss function. In particular, we will solve a series of C optimization problem with each optimization problem written as follows:

$$\begin{aligned} \min_{S_{k,1}, \cdots, S_{k,C}} \quad & \sum_{l=1}^{C} \sum_{i=1}^{C} w_i (S_{k,l} - [S_i]_{k,l})^2 \\ \text{s.t.} \quad & S_{k,l} \geq 0, \ l = 1, 2, \cdots, C \\ & \sum_{l=1}^{C} S_{k,l} = 1. \end{aligned} \tag{4.20}$$

This is a quadratic programming problem. To see more clearly about the meaning of the result, we redefine the problem as follows:

$$\begin{aligned} \min_{\mathbf{s}} \quad & \sum_{i=1}^{n} w_i \|\mathbf{s} - \mathbf{s}_i\|_2^2 \\ \text{s.t.} \quad & \mathbf{s} \succeq 0, \mathbf{e}^\top \mathbf{s} = 1. \end{aligned} \tag{4.21}$$

where $\mathbf{s}_k = (S_{k,1}, S_{k.2}, \cdots, S_{k,C})$ and $\mathbf{s}_i^k = ([S_i]_{k,1}, [S_i]_{k,2}, \cdots, [S_i]_{k,C})$. The dual form of the above optimization problem can be further written as

$$\begin{aligned} \max_{\lambda,\gamma} \quad & \lambda - \hat{\mathbf{s}}_k^\top (\lambda \mathbf{e} + \gamma) - \frac{\|\gamma + \lambda \mathbf{e}\|_2^2}{2\sum_{i=1}^{n} w_i} \\ \text{s.t.} \quad & \gamma \succeq 0, \ \lambda \geq 0. \end{aligned} \tag{4.22}$$

where $\hat{\mathbf{s}}_k$ is defined as

$$\hat{\mathbf{s}}_k = \frac{\sum_{i=1}^{n} w_i \mathbf{s}_i^k}{\sum_{i=1}^{n} w_i} \tag{4.23}$$

The solution $\mathbf{s}_k$ is calculated as

$$\mathbf{s}_k = \hat{\mathbf{s}_k} + \frac{\gamma + \lambda \mathbf{e}}{\sum_{i=1}^{n} w_i} \tag{4.24}$$

As we can see, the two constraints $\mathbf{s} \succeq 0$ and $\mathbf{e}^\top \mathbf{s} = 1$ result in the additional term $\frac{\gamma + \lambda \mathbf{e}}{\sum_{i=1}^{n} w_i}$.

l_2 loss function It is defined as $\max_{\|\mathbf{z}\|_2=1} \mathbf{z}^\top (S - S')^\top (S - S')\mathbf{z}$. This loss function essentially requires the two transitional matrix S and S' to be similar along any direction $\mathbf{z}$. The interesting point is that the first loss function l_1 can also be written in the similar form to l_2 as $l_1(S, S') = \mathbf{e}^\top (S - S')^\top (S - S')\mathbf{e}$. In other words, the first loss function only require the two matrices to be matched along the direction $\mathbf{e}$. Thus, $l_2(S, S')$ is a more general form than $l_1(S, S')$.

l_2 loss function leads to the following optimization problem

$$\begin{aligned} \min_{S} \quad & \max_{|\mathbf{z}|\leq 1} \mathbf{z}^\top \Big(\sum_{i=1}^{n} w_i (S - S_i)^\top (S - S_i) \Big) \mathbf{z} \\ \text{s.t.} \quad & S_{k,l} \geq 0,\ k, l = 1, 2, \cdots, C \\ & S^\top \mathbf{e} = \mathbf{e}. \end{aligned} \tag{4.25}$$

Since $\max_{|\mathbf{z}|\leq 1} \mathbf{z}^\top \Big(\sum_{i=1}^{n} w_i (S-S_i)^\top (S-S_i) \Big) \mathbf{z}$ is the principle eigenvalue of matrix $\sum_{i=1}^{n} (S - S_i)^\top (S - S_i)\mathbf{z}$, we can rewrite the above problem into the following format:

$$\begin{aligned} \min_{S,t,A_i} \quad & t \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i A_i \leq t I_C \\ & \begin{pmatrix} A_i & (S - S_i)^\top \\ S - S_i & I \end{pmatrix} \succeq 0 \\ & S_{k,l} \geq 0,\ k, l = 1, 2, \cdots, C \\ & S^\top \mathbf{e} = \mathbf{e}. \end{aligned} \tag{4.26}$$

where I_C and I are both identity matrix.

As we mentioned above, l_1 loss function can be viewed as a special case of l_2 loss function and it can be verified through the above optimization problem. To better understand this, consider the special case where $A_i = diag(a_i^1, a_i^2, \cdots, a_i^C)$. Furthermore, we approximate the constraint

$$\begin{pmatrix} A_i & (S - S_i)^\top \\ S - S_i & I \end{pmatrix} \succeq 0 \tag{4.27}$$

by its necessary conditions, i.e.,

$$a_i^k \geq \|\mathbf{s}_k - \mathbf{s}_i^k\|_2^2$$

As the result of the above simplification, we rewrite the original problem as

$$\begin{aligned} \min_{t,\mathbf{s}_k} \quad & t \\ \text{s. t.} \quad & \sum_{i=1}^{n} w_i a_i^k \leq t,\ k = 1, 2, \cdots, C \\ & a_i^k \geq \|\mathbf{s}_k - \mathbf{s}_i^k\|_2^2,\ k = 1, \cdots, C,\ i = 1, \cdots, n \\ & \mathbf{s}_k \succeq 0,\ k = 1, 2, \cdots, C \\ & \mathbf{s}_k^\top \mathbf{e} = 1,\ k = 1, 2, \cdots, C. \end{aligned} \tag{4.28}$$

To minimize t, it is sufficient to minimize $\sum_{i=1}^{n} w_i a_i^k$. It is not too hard to see that this is equivalent to the optimization problem using l_1 loss function.

4.3.4 Case Study: Multi-label Categorization

We evaluate the proposed ranking propagation scheme on multiple data sets and report the empirical results in the following study. We will answer the following questions:

- Does *Rank Propagation* really help the multi-label classification task? As we discussed before, the traditional methods for multi-label classification have problems caused by utilizing class labels as numerical values. *Rank Propagation* utilizes the preferences among all categories and can avoid this problem.

- Does $l2$ loss function work better than $l1$ loss function? $l2$ loss function considers the projection of the difference between the preference matrix of the test example in question and the preference matrix of all training examples along all directions. Theoretically, the optimization problem based $l2$ loss function should result in better performance and the empirical studies proved this.

- How sensitive is *Rank Propagation* to the parameters? There are two parameters involved in *Rank Propagation*: A and B. We give a detailed description of the influence of these parameters on the performance.

Datasets

We evaluate our proposed approach with three datasets: *MovieLens* dataset, *St. Andrews* dataset and *Yeast* dataset. In three datasets, each document is assigned to one or more categories. We describe the datasets as follows:

- *MovieLens* C18 dataset: *MovieLens* dataset is a movie rating dataset. The original dataset provides the ratings of 943 users on 1682 movies from 19 categories. Each rating is an integer ranging from 1 to 5 with 1 being the lowest rating and 5 is the highest rating. Zero rating means that the rating information is not available. We selected 849 movies and 18 categories as our testbed. The average number of cat-

egories per movie is around 3 and the average number of movies per categories is around 114 in the resulting dataset.

- *St. Andrews* C10 dataset: This is an image categorization dataset which contains 999 predefined categories and 28133 images along with their captions (textual description). The 28133 captions consist of 44085 terms and 1348474 word occurrences. The maximum caption length is 316 words, but on average 48 words in length. We selected 10 categories and 1000 images with their captions as our text categorization testbed (we view each caption as a document). The number of categories per document in the resulting dataset varies from 2 to 4 and the average number is around 2. The number of documents for each category varies from 108 to around 320, and the average number of documents per category is around 209.

- *Yeast* C14 dataset: The *Yeast* dataset is formed by micro-array expression data and phylogenetic profiles with 2417 genes. East gene is represented by 103 attributes and is associated with a set of functional classes whose maximum size can be potentially more than 190. The dataset in our experiment contains 14 classes. The number of classes per gene is ranging from 1 to 11 with the average number being around 5. The average number of genes per class is ranging from 34 to 1816 with the average number being around 732.

Baselines

The proposed Rank Propagation method is a supervised-learning algorithm which predicts the labels by propagating the preferences between labeled data and unlabeled data. We refer to the Rank Propagation approaches using two loss functions as RP L1 and RP L2 respectively. We compare the Rank Propagation approaches with two other supervised-learning algorithms: K Nearest Neighbor method (KNN) and Support Vector machine (SVM) as our baselines.

KNN is similar to the Rank Propagation method except that KNN propagates the *true labels* of the training examples while Rank Propagation propagates the *preferences* among categories which are obtained from the given labels of the training examples. In order to generate a ranking list of all categories for each example using KNN, we conduct KNN for each category and generate a ranking list of all categories based on the resulting scores.

Traditional SVM is often used for binary classification tasks. Many algorithms derived from SVM have been proposed for multi-label classification tasks. Among them, SVM binary appears to perform the best according to the empirical results in [33]. The basic idea of SVM binary can be summarized as two steps. First, we conduct binary classification for each category and generate a score for each testing example on each category. For a testing example, the score corresponding to one category can be viewed as a confidence score indicating how likely the testing example should be assigned to this category. Second, we generate a ranking list of all categories according to the confidence scores for each testing example. The higher rank a category has, the more likely the testing example should be assigned to this category. Various metrics can be applied to the ranking lists in order to evaluate the performance.

Evaluation Metrics

In the framework we describe in section 4.3.3, we generate a ranking list of all categories for each document. Similarly, we generate a ranking list of all categories for each testing example in two baseline algorithms. Thus, it's more appropriate to evaluate the performance of three algorithms using rank-related measurements. In our empirical study, we use category-wise *Precision/Recall-breakeven point* (PRBEP), *Micro-average F1 score* and *Macro-average F1 score* at different ranks, and *Area under ROC curve* as our evaluation metrics.

For *Micro-average F1 score*, we first compute the precision and recall at each rank of category for each document, then compute the average precision and recall at each

rank across all documents and finally compute F1 score based on the precision and recall. *Macro-average F1 score* is computed similarly, but the precision and recall are computed for each category.

The *Precision/Recall-breakeven point* is a commonly used measure for evaluating text classifiers. It is based on the two statistics *recall* and *precision* which are both widely used in information retrieval. Between high recall and high precision exists a trade-off. The *Precision/Recall-breakeven point* is defined as that value for which precision and recall are equal. We compute the average *Precision/Recall-breakeven point* value across all categories as follows: first, for each category, we compute the precision/recall when assigning the top n examples to this category and n is the number of examples actually in this category; second, we take the average precision/recall across all categories as the final results.

ROC (Receiver Operating Characteristic) curve was first used to define detection cut-off points for radar equipment with different operators. It is also a commonly used measure in information retrieval. In ROC curve plot, sensitivity (true positive rate) is plotted against 1-specificity (false positive rate). To get a comprehensive idea of the ranking list, we compute the average *Area under ROC curve*. First, we compute the true positive rate and false positive rate at each ranks for each document and plot the ROC curve for each example. Then we compute the area under ROC curve for each example and take the average as the final results.

By using the above three metrics, we can get a comprehensive idea about the ranking lists of categories generated in three algorithms. We conduct each experiment 10 times and take the average results over 10 trials as the final results.

Parameter Setup

There are two parameters involved in the Ranking Propagation method: A and B (see details in section 4.3.3). Since only the ratio of A and B matters, we fix $A = 1$ and test

different B values. We set $B = 10$ which gives the best performance for all three experiments. We'll include the discussion on the influence of B values in the following sections. We also need to compute the similarity matrix among data examples in the Ranking Propagation method. Each entry in the similarity matrix is computed using cosine similarity measure based on the given representation of the data points.

We download SVM-light ([20]) as the implementation of SVM and use the default setting of the software. For KNN, K (the number of the nearest neighbors) was set to be empirical value generating the best results.

Results and Analysis I: Comparison among All Methods

We evaluate the effectiveness of the proposed Ranking Propagation method in the first experiment. Table 4.1 shows the *Precision/Recall-breakeven point* for the three datasets described in the previous sections. Table 4.2, 4.3 and 4.4 presented the *Area under ROC Curve* for three datasets with different number of training examples. Also, Figure 4.5, 4.4 and 4.5 showed the *Micro-average* and *Macro-average F1 scores* of three datasets. We have the following observations.

First, for all the algorithms, as the number of training examples increases, the performance is also getting better in terms of all three metrics. Moreover, the curve of *F1 scores* usually starts at a low value, then keeps ascending till it reaches a peak and finally ends at a fixed point lower than the peak value. This is consistent with the intuition since it is hard to make all correct predictions at the first rank. But as the rank increases, it is more possible to catch the positive examples for each category. The ascending trend is continued until at a certain point there are not many positive examples left for each category at lower ranks. Then the curve starts descending until it reaches the end. There many be some fluctuation in a curve because some positive examples are not ranked higher than the negative ones.

Second, we can see that in most cases, both RP L1 and RP L2 outperform two baselines, namely, SVM binary and KNN. This indicates that propagating the preferences instead of

true labels does help the classification in some applications. RP L2 method shows significant improvement in terms of all the metrics. This is not too hard to understand since RP L2 considers a more generalized form of differences and constructs a better optimization problem.

Third, it is interesting that KNN actually outperforms SVM binary method in some cases despite that KNN is a simple algorithm. For instance, KNN achieves the better *Micro-average* and *Macro-average* F1 scores for *yeast* datssets in Figure 4.5 than SVM binary. This may be related to the nature of the datasets including the distribution of the positive examples of each class, the numerical presentation of the data examples, etc.

Table 4.1: PRBEP for Different Datasets with Different Number of Training Examples.

	Number of Training examples			
Classifier	20	40	80	100
RP L1	0.135 ± (0.008)	0.138 ± (0.006)	0.150 ± (0.006)	0.162 ± (0.010)
RP L2	0.136 ± (0.006)	0.144 ± (0.005)	0.168 ± (0.003)	0.184 ± (0.002)
SVM	0.131 ± (0.007)	0.136 ± (0.006)	0.148 ± (0.002)	0.152 ± (0.003)
KNN	0.132 ± (0.006)	0.133 ± (0.006)	0.133 ± (0.007)	0.135 ± (0.007)

(a) *MovieLens* Dataset

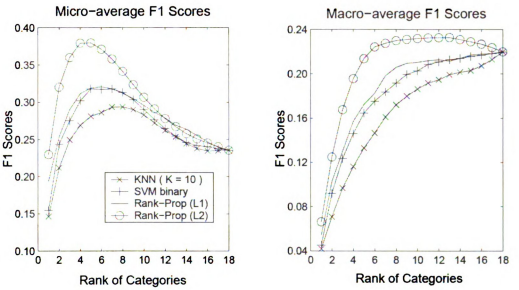
	Number of Training examples			
Classifier	100	200	300	400
RP L1	0.211 ± (0.007)	0.213 ± (0.006)	0.213 ± (0.003)	0.214 ± (0.006)
RP L2	0.217 ± (0.011)	0.224 ± (0.005)	0.237 ± (0.004)	0.243 ± (0.009)
SVM	0.206 ± (0.011)	0.207 ± (0.009)	0.209 ± (0.009)	0.211 ± (0.008)
KNN	0.208 ± (0.004)	0.212 ± (0.005)	0.2112 ± (0.006)	0.212 ± (0.007)

(b) *St. Andrews* C-10 Dataset.

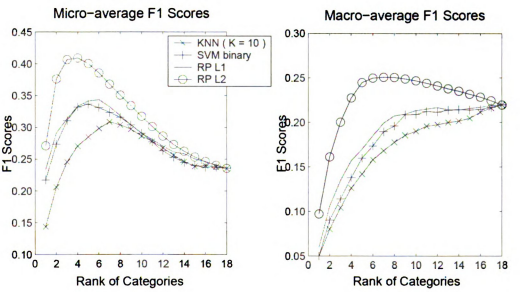
	Number of Training examples			
Classifier	300	500	1000	1500
RP L1	0.303 ± (0.004)	0.308 ± (0.003)	0.309 ± (0.003)	0.315 ± (0.005)
RP L2	0.316 ± (0.004)	0.336 ± (0.009)	0.352 ± (0.001)	0.390 ± (0.009)
SVM	0.301 ± (0.006)	0.302 ± (0.005)	0.304 ± (0.002)	0.306 ± (0.003)
KNN	0.301 ± (0.006)	0.301 ± (0.005)	0.302 ± (0.009)	0.303 ± (0.008)

(c) *Yeast* C-14 Dataset

Figure 4.3: F1 Scores of Three Algorithms for *MovLens* C18 Dataset.

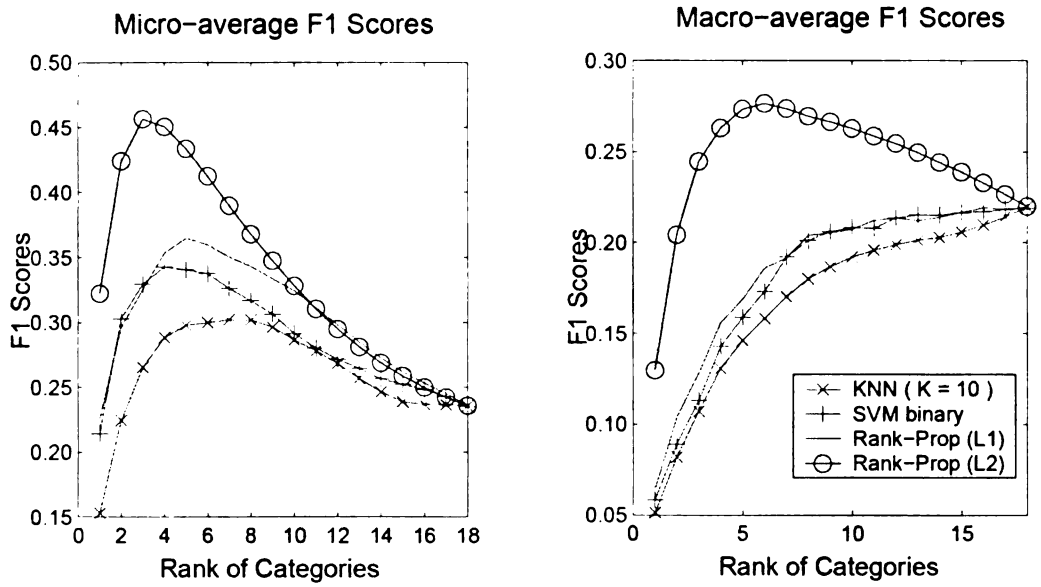


(a) The Number of Training data = 20

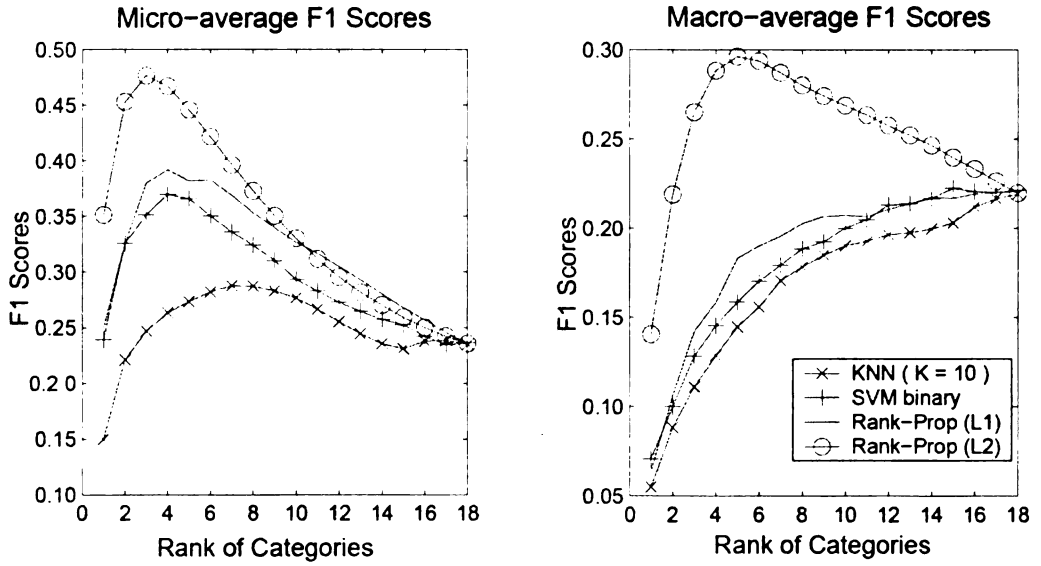


(b) The Number of Training data = 40

Figure 4.4: F1 Scores of Three Algorithms for *MovLens* C18 Dataset (cont'd).

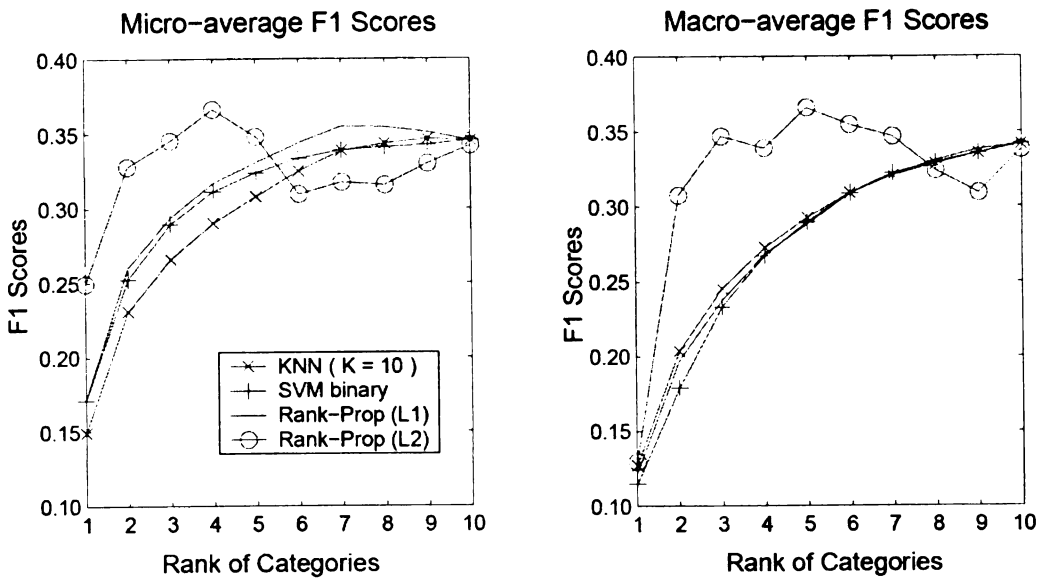


(c) The Number of Training data = 80

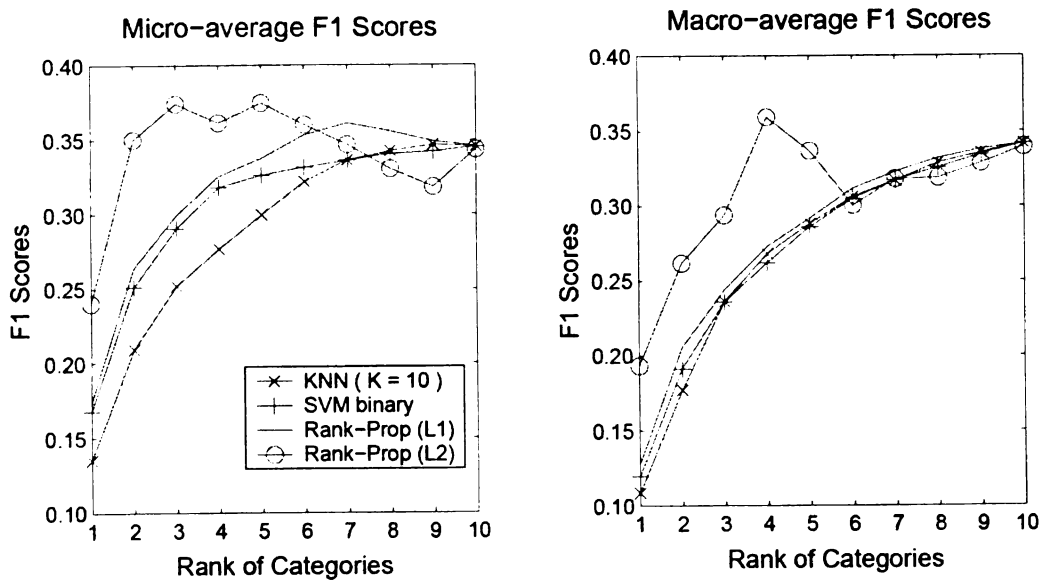


(d) The Number of Training data = 100

Figure 4.4: F1 Scores of Three Algorithms for *St. Andrews* C10 Dataset.

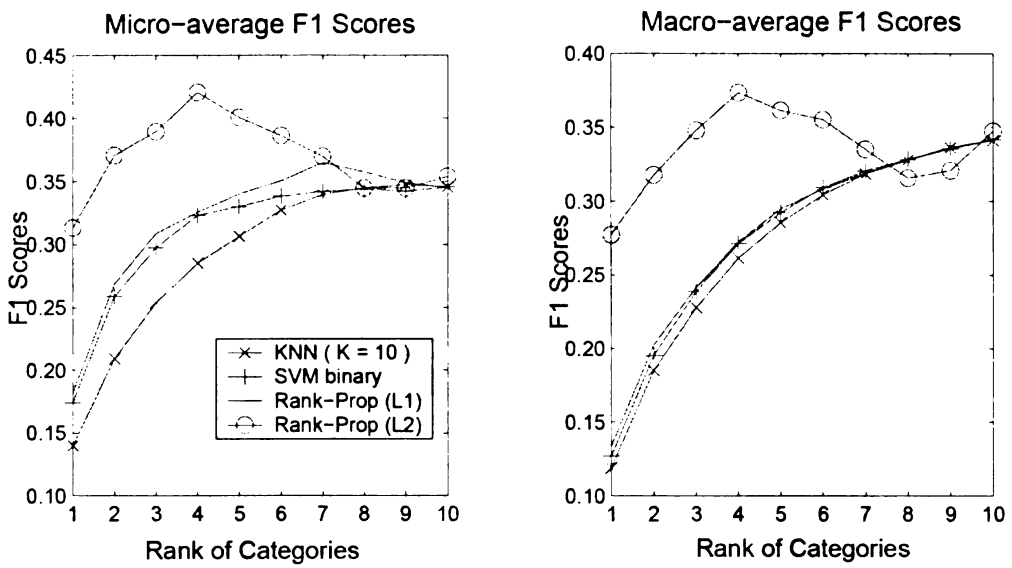


(a) The Number of Training data = 100

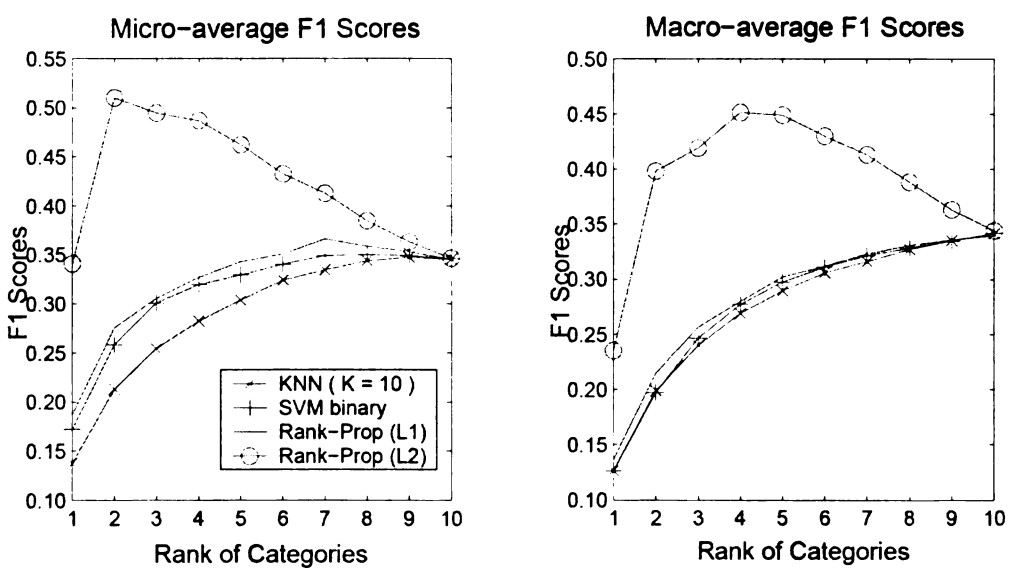


(b) The Number of Training data = 200

Figure 4.5: F1 Scores of Three Algorithms for *St. Andrews* C10 Dataset (cont'd).

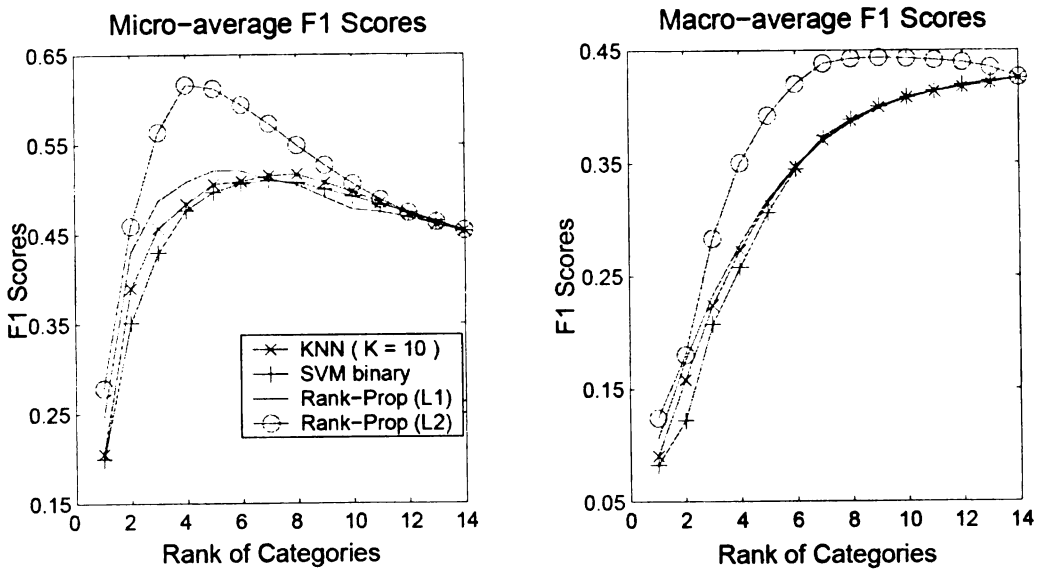


(c) The Number of Training data = 300

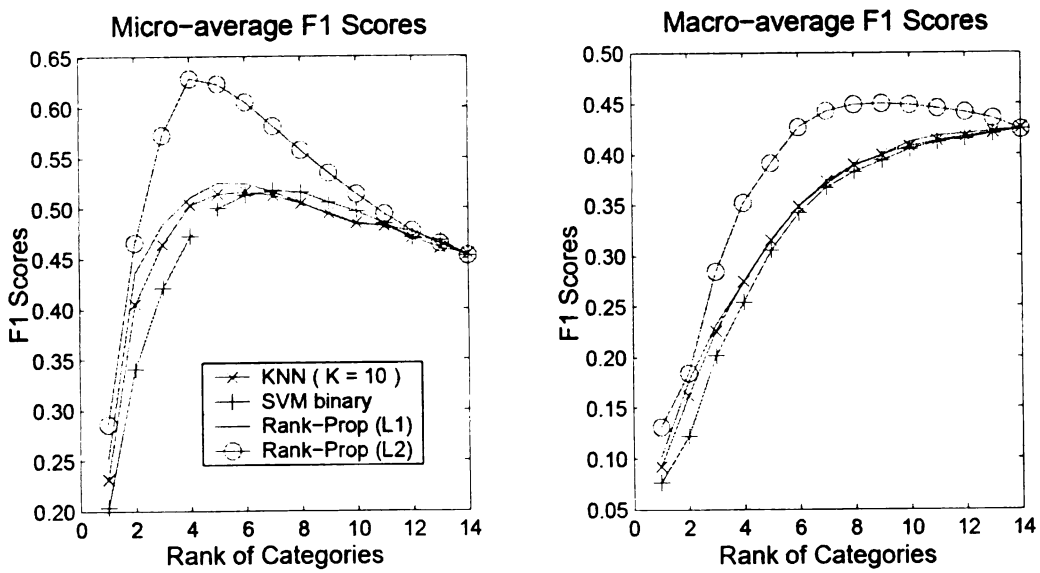


(d) The Number of Training data = 400

Figure 4.5: F1 Scores of Three Algorithms for *Yeast* C14 Dataset.

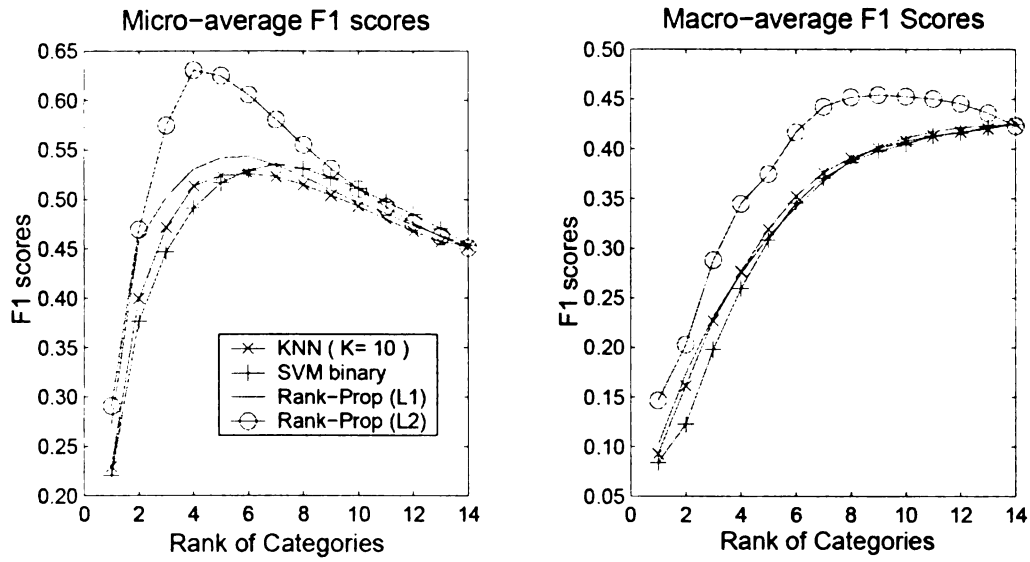


(e) The Number of Training data = 300

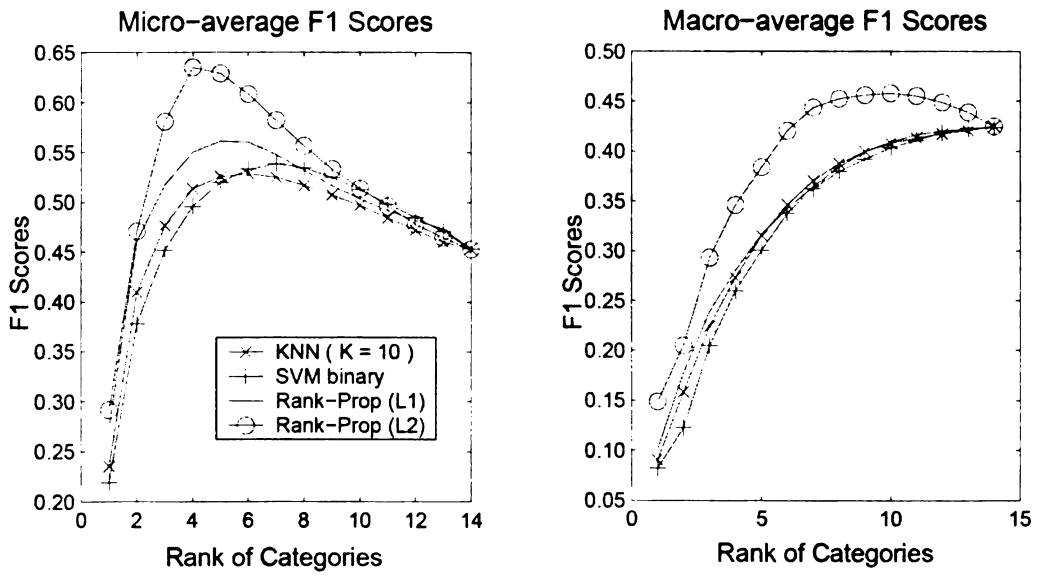


(f) The Number of Training data = 500

Figure 4.5: F1 Scores of Three Algorithms for *Yeast* C14 Dataset (cont'd).



(g) The Number of Training data = 1000



(h) The Number of Training data = 1500

Table 4.2: Area under ROC Curve for *MovieLens* Dataset.

	Number of Training examples			
Classifier	20	40	80	100
RP L1	0.640 ± (0.012)	0.663 ± (0.009)	0.690 ± (0.025)	0.728 ± (0.003)
RP L2	0.714 ± (0.025)	0.742 ± (0.002)	0.779 ± (0.011)	0.790 ± (0.008)
SVM	0.600 ± (0.008)	0.628 ± (0.031)	0.631 ± (0.008)	0.682 ± (0.007)
KNN	0.612 ± (0.029)	0.625 ± (0.062)	0.636 ± (0.040)	0.641 ± (0.030)

(a) Micro-average

	Number of Training examples			
Classifier	20	40	80	100
RP L1	0.591 ± (0.014)	0.613 ± (0.004)	0.643 ± (0.009)	0.655 ± (0.007)
RP L2	0.608 ± (0.022)	0.628 ± (0.024)	0.656 ± (0.019)	0.665 ± (0.012)
SVM	0.575 ± (0.015)	0.599 ± (0.005)	0.611 ± (0.015)	0.623 ± (0.007)
KNN	0.569 ± (0.005)	0.579 ± (0.003)	0.597 ± (0.007)	0.608 ± (0.004)

(b) Macro-average

Table 4.3: Area under ROC Curve for *St. Andrews* Dataset.

	Number of Training examples			
Classifier	100	200	300	400
RP L1	0.641 ± (0.015)	0.644 ± (0.011)	0.645 ± (0.010)	0.651 ± (0.011)
RP L2	0.650 ± (0.009)	0.654 ± (0.010)	0.672 ± (0.008)	0.743 ± (0.007)
SVM	0.634 ± (0.021)	0.636 ± (0.007)	0.638 ± (0.008)	0.649 ± (0.007)
KNN	0.512 ± (0.011)	0.516 ± (0.009)	0.519 ± (0.009)	0.524 ± (0.009)

(a) Micro-average

	Number of Training examples			
Classifier	20	40	80	100
RP L1	0.553 ± (0.023)	0.592 ± (0.010)	0.596 ± (0.011)	0.596 ± (0.013)
RP L2	0.582 ± (0.001)	0.621 ± (0.003)	0.665 ± (0.029)	0.697 ± (0.014)
SVM	0.533 ± (0.019)	0.578 ± (0.011)	0.581 ± (0.008)	0.584 ± (0.013)
KNN	0.519 ± (0.011)	0.522 ± (0.010)	0.527 ± (0.010)	0.529 ± (0.009)

(b) Macro-average

Table 4.4: Area under ROC Curve for *Yeast* Dataset.

	Number of Training examples			
Classifier	300	500	1000	1500
RP L1	0.690 ± (0.010)	0.722 ± (0.0041)	0.734 ± (0.010)	0.754 ± (0.008)
RP L2	0.715 ± (0.013)	0.754 ± (0.002)	0.784 ± (0.005)	0.797 ± (0.008)
SVM	0.648 ± (0.011)	0.710 ± (0.006)	0.721 ± (0.005)	0.727 ± (0.009)
KNN	0.703 ± (0.009)	0.709 ± (0.005)	0.710 ± (0.006)	0.715 ± (0.004)

(a) Micro-average

	Number of Training examples			
Classifier	300	500	1000	1500
RP L1	0.594 ± (0.010)	0.599 ± (0.002)	0.599 ± (0.003)	0.612 ± (0.009)
RP L2	0.598 ± (0.008)	0.606 ± (0.003)	0.607 ± (0.003)	0.621 ± (0.004)
SVM	0.589 ± (0.003)	0.589 ± (0.004)	0.598 ± (0.006)	0.601 ± (0.002)
KNN	0.586 ± (0.004)	0.587 ± (0.004)	0.588 ± (0.007)	0.600 ± (0.004)

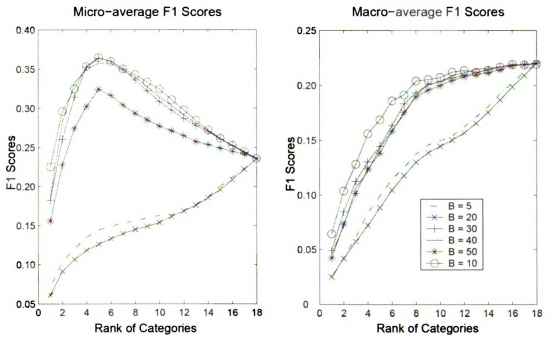
(b) Macro-average

Results and Analysis II: Sensitivity Test

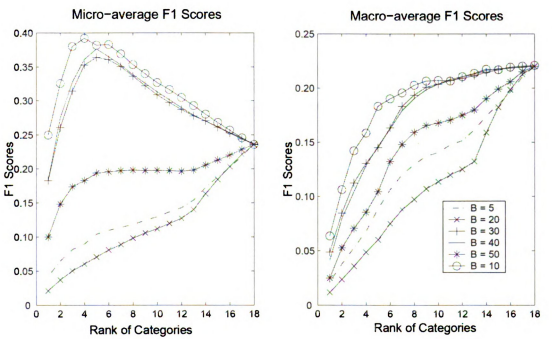
We discuss the influence of the parameters A and B on the proposed Rank Propagation algorithm in this empirical study. The following are the F1 scores for *Yeast* dataset with 80 and 100 training examples for different B values (since only the ratio between A and B matters, we fix $A = 1$ and consider the value of B).

As we can see, values smaller or greater than the optimal value for B all result in worse performance than the optimal value. It is consistent with the intuition since too large values will dominate the resulting propagation scores and too small values can not express the differences in transitional probabilities. The parameter B has the similar influence on RP L2 to RP L1, thus we didn't include it here.

Figure 4.5: F1 scores of Rank Propagation algorithm using $l1$ loss function with different B values on *MovLens* C18 dataset.



(a) The Number of Training data = 80



(b) The Number of Training data = 100

Chapter 5

Propagation over Directed Graph

Label propagation approaches have gained popularity in semi-supervised learning area. Most previous research has been done on the undirected graph [4, 5, 57, 55, 24, 51]. However, there are many scenarios which involve directed graphs. For example, in order to explore the link structure of the web for ranking web pages, we usually view the links between web pages as directed. A number of studies have been devoted to semi-supervised learning on the directed graph such as [50, 52]. In this chapter, we will explore the label propagation approach over directed graphs in the application of *text categorization*.

Label propagation has been shown to be an effective approach for text categorization [57, 23]. The main idea is to predict the category assignments for unlabeled documents by propagating the category information of the training documents through the similarities between documents. Most previous research on label propagation focuses on undirected graphs which are based on symmetric similarity matrices. However, in a number of scenarios, the similarity relationship between documents can be asymmetric and is better captured by directed graphs. For example, consider two documents d_A and d_B with document d_A being a part of document d_B. Evidently, if document d_A belongs to category c, we would expect that document d_B should also belong to category c. However, this inference is irreversible, namely we can not infer the category for document d_A if we know the category

of d_B. To capture the asymmetric relationship among documents, we propose the label propagation approach for text categorization using directed graphs.

The proposed approach first constructs a directed graph from document collection, then converts this directed graph into an undirected one, and finally makes prediction by propagating the class labels over the converted undirected graph. There are two questions which need to be answered:

How do we utilize the directed graph? A straightforward way is to first convert the direct graph into an undirected one and then apply a standard label propagation method in this converted undirected graph. In our study, we will discuss the work in [52]. It proposed a regularization framework which converts a directed graph based on asymmetric similarity matrix into an undirected graph. We then propagate the labeling information on the undirected graph using *Spectral Graph Transducer* [23].

How do we construct a directed graph? To construct a directed graph from document collection, it requires computing asymmetric similarities, namely similarity of d_A to d_B is different from the similarity of d_B to d_A. Two asymmetric similarity measures are presented in this study, i.e., the asymmetric similarity based on the KL divergence and the asymmetric similarity based on the modified cosine similarity. We evaluate the efficacy of these two asymmetric similarities through an empirical study with multiple datasets.

5.1 Convert a Directed Graph into an Undirected Graph

Let $\mathcal{D} = (\mathbf{d}_1, \cdots, \mathbf{d}_n)$ denote the entire collection of documents. Assume the first n_l documents of $\mathcal{D}$ are labeled by $\mathbf{y}_l = (y_1, \cdots, y_{n_l})$ and the remaining $n_u = n - n_l$ documents are unlabeled. Each class label y_i is either $+1$ or -1. Let $w(i\|j) \geq 0$ denote the similarity of the ith example to the jth example. Note that the similarity is asymmetric, namely $w(i\|j) \neq w(j\|i)$. We further denote by $W \in \mathbb{R}_+^{n \times n}$ the similarity matrix for all the exam-

ples. Our goal is to predict $\mathbf{y}_u$, the category labels for the unlabeled documents, by propagating the class labels $\mathbf{y}_l$ over the asymmetric similarities $w(i\|j)$. In the following, we will first describe the approach in [50] that is used to convert a directed graph into an undirected graph. We will then describe the spectral graph transducer (SGT) algorithm [23], which is used to propagate the class labels over the converted undirected graph.

5.1.1 Conversion to Undirected Graphs

Given a set of labeled documents $\mathcal{D}_l = \{(\mathbf{d}_1, c_1), \cdots, (\mathbf{d}_l, c_l)\}$, the task is to classify the unlabeled documents $\mathcal{D}_u = \{\mathbf{d}_{l+1}, \cdots, \mathbf{d}_{l+u}\}$. Assume $n = l + u$. We can construct a directed graph $G = (V, E)$ as described in the previous section. V is the set of nodes and each node represents a document. E is the set of edges. The edges are weighted and there is a weight function $w : E \rightarrow \mathbb{R}^+$ which associates a positive value $w([u, v])$ with each edge $[u, v] \in E$. We use K-L divergence between two documents based on their term vectors as the weight function. The *out-degree* function $d+ : V \rightarrow \mathbb{R}^+$ and *in-degree* function $d^- : V \rightarrow \mathbb{R}^-$ are defined as: $d^+(u) = \sum_{u \rightarrow v} w([u, v])$ and $d^-(u) = \sum_{u \leftarrow v} w([u, v])$. $u \rightarrow v$ denotes the set of vertices adjacent from the vertex u and $u \leftarrow v$ denotes the set of vertices adjacent to u.

For a given weighted directed graph, there is a natural random walk on the graph with the transition probability function $p : V \times V \rightarrow \mathbb{R}^+$ defined by $p(u, v) = w([u, v])/d^+(u)$ for all $[u, v] \in E$, and 0 otherwise. The graph we constructed is strongly connected and aperiodic and the random walk has a unique *stationary distribution* π which satisfies the *balance equations* $\pi(v) = \sum_{u \rightarrow v} \pi(u)p(u, v)$, for all $v \in V$. $\pi(v) > 0$ for all $v \in V$.

Assume a classification function $f \in \mathcal{H}(V)$, which assigns a label sign $f(v)$ to each vertex $v \in V$. A functional Ω can be defined as

$$\Omega(f) := \frac{1}{2} \sum_{[u,v] \in E} \pi(u)p(u, v) \left(\frac{f(u)}{\sqrt{\pi(u)}} - \frac{f(v)}{\sqrt{\pi(v)}} \right)^2 \tag{5.1}$$

Ω sums the weighted variation of a function on each edge of the directed graph. The initial label assignment should be be changed as little as possible. Let y denote the function in $\mathcal{H}(V)$ defined by $y(v) = 1$ or -1 if vertex v has been labeled as positive or negative respectively, or 0 if it's unlabeled. We can form the optimization problem as follows:

$$\text{argmin}_{f \in \mathcal{H}(V)} \{\Omega(f) + \mu \parallel f - y \parallel^2\} \tag{5.2}$$

where $\mu > 0$ is the parameter specifying the tradeoff between the two competitive terms.

Define the operator $\Theta : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$ as follows:

$$(\Theta f)(v) = \frac{1}{2}\left(\sum_{u \rightarrow v} \frac{\pi(u)p(u,v)f(u)}{\sqrt{\pi(u)\pi(v)}} + \sum_{u \leftarrow v} \frac{\pi(v)p(v,u)f(u)}{\sqrt{\pi(v)\pi(u)}}\right)$$

Θ can be written in matrix form as

$$\Theta = \frac{\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P^{\top} \Pi^{1/2}}{2} \tag{5.3}$$

where Π denote the diagonal matrix with $\Pi(v,v) = \pi(v)$ for all $v \in V$. P is the transition probability matrix and $P^{\top}$ is the transpose of P. **Note**: Θ is symmetric although the original weight matrix is asymmetric.

Using Θ as the new symmetric similarity matrix, we conduct the propagation using SGT [23] which is discussed in details in the following section. Figure 5.1 shows the algorithm. **Note** that for multi-class problem, we compute the function f for each category using the algorithm.

5.1.2 Propagation Scheme: Spectral Graph Transducer

We briefly introduce Spectral Graph Transducer (SGT) in this section. SGT is a method for transductive learning which can be seen as a transductive version of the k nearest-neighbor classifier. In this method, the training problem has a meaningful relaxation that can be

solved globally optimally using spectral methods. A key advantage of the method is that it doesn't require additional heuristics to avoid unbalanced splits which makes this algorithm very robust.

Let $\mathbf{y} = (y_1, \ldots, y_n)$ denote the class labels for all the examples. Evidently, $\mathbf{y}_u = (y_{n_l+1}, \ldots, y_n)$. Given a symmetric similarity matrix S, we need to find the class labels $\mathbf{y}$ that on one hand is consistent with $\mathbf{y}_l$, and on the other hand is consistent with the similarity matrix S. Follow [23], we can formulate the above objectives into the following optimization problem,

$$\begin{aligned} \min_{\mathbf{y}\in\mathbf{R}^n} \quad & \sum_{i,j=1}^{n} S_{i,j}(y_i - y_j)^2 + c\sum_{i=1}^{n_l}(y_i - \gamma_i)^2 \\ \text{s. t.} \quad & \sum_{i=1}^{n} y_i = 0, \sum_{i=1}^{n} y_i^2 = n \end{aligned} \tag{5.4}$$

where γ_i is γ_+ if $y_i = +1$ and γ_- otherwise. [23] computes the similarity-weighted k

Input: The label assignment vector $\mathbf{y}$ containing the labels for training data and 0 for testing data; the similarity matrices $S^{\mathcal{D}}$ for documents computed by K-L divergence.
Output: The classification function f.

Algorithm:
1. Construct the directed graph: construct the graph $G = \langle V, E\rangle$. V is the set of nodes and each node represents a document. E is the set of edges and each edge $[u, v]$ is assigned a weight $w([u, v])$ determined by $S^{\mathcal{D}}(u, v)$. Compute the transition probability matrix P defined as $p(u, v) = w([u, v])/d^+(u)$.

1. Define the random walk: define a random walk over G with a transition matrix P such that it has a unique stationary distribution π.

2. Compute the matrix Θ: Let Π denote the diagonal matrix with its diagonal elements being the stationary distribution π of the random walk. Compute the matrix Θ as $\Theta = (\Pi^{1/2}P\Pi^{-1/2} + \Pi^{-1/2}P^{\top}\Pi^{1/2})/2$.

3. Compute the function f: Compute the function f using SGT, and classify each unlabeled vertex v as sign $f(v)$.

Figure 5.1: The Algorithm for Classification on a Directed Graph

nearest-neighbor graph S as

$$S'_{ij} = \begin{cases} \frac{sim(\vec{x}_i, \vec{x}_j)}{\sum_{\vec{x}_k \in knn(\vec{x}_i)} sim(\vec{x}_i, \vec{x}_k)} & \text{if } \vec{x}_j \in knn(\vec{x}_i) \\ 0 & \text{else} \end{cases}$$

and then symmetricize it as $S = S + S'$. The details of computing $\gamma_{\pm}$ can be found in [23].

5.2 Construct a Direct Graph

Label propagation approaches often involve a connected graph constructed from the data samples. Consider a classification problem. Assume we have the labeled data set $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$ and the unlabeled data set $\{\mathbf{x}_{l+1}, \cdots, \mathbf{x}_{l+u}\}$. We construct a connected graph $G = (V, E)$: V is the set of nodes in which each node represents one data sample; E is the set of edges between nodes and each edge is assigned with a value which is determined by the weight function w. The function w is usually computed based on the similarity between all nodes. For instance, we use RBF kernel to compute the similarity. If w is a symmetric measure which means $w(x_i, x_j) = w(x_j, x_i)$, the graph is an undirected graph since the edge from the node x_i to x_j is viewed equally as the edge from the node x_j to x_i. All the approaches we discussed from the previous chapters are based on undirected graph. In many cases, directed graphs may be a more appropriate choice to express the nature of the data set. By defining the weight function w as an asymmetric function, we can construct a directed graph.

Let's consider the document classification problem. Given a set of labeled documents $\mathcal{D}_l = \{(\mathbf{d}_1, c_1), \cdots, (\mathbf{d}_l, c_l)\}$, the task is to classify the unlabeled documents $\mathcal{D}_u = \{\mathbf{d}_{l+1}, \cdots, \mathbf{d}_{l+u}\}$. In order to apply the label propagation approach, we need to compute the document similarity matrix S. Two asymmetric similarity measures are presented in the following study, i.e., the asymmetric similarity based on the K-L divergence and the asymmetric similarity based on modified cosine similarity.

K-L divergence similarity measure For probability distribution P and Q of a discrete variable, the K-L divergence of Q from P is defined to be

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{5.5}$$

Given the document $\mathbf{d}_i$ and the document $\mathbf{d}_j$, we first compute the pairwise KL distance $D(i\|j)$ as follows:

$$D(i\|j) = \sum_{k=1}^{m} \tilde{d}_{i,k} \log \left(\frac{\tilde{d}_{i,k}}{\tilde{d}_{j,k}} \right) \tag{5.6}$$

where $\tilde{d}_{i,k}$ is normalized term frequency and is defined as $\tilde{d}_{i,k} = d_{i,k} / \sum_{k'=1}^{m} d_{i,k'}$. We then convert the distance $D(i\|j)$ to similarity $w(i\|j)$ by $w(i\|j) = \exp(-\lambda D(i\|j))$ where λ is a scaling factor and is determined empirically.

The graph is directed since the edge from the node representing $\mathbf{d}_i$ to the node representing $\mathbf{d}_j$ is different from the edge from the node representing $\mathbf{d}_j$ to the node representing $\mathbf{d}_i$. By using KL divergence, we are trying to express in the pairwise similarities the differences from document length, term probability distribution and some other factors.

Asymmetric cosine similarity We assume that the document length reflects the variety of topics which may have impact on propagation. Thus the resulting similarity should be weighted according the document length. Standard cosine similarity is a symmetric measure. To capture the differences from document lengths, we modify it to be a asymmetric measure.

Let's denote by $\mathcal{X}_i$ the set of words used by the ith document, i.e., $\mathcal{X}_i = \{j | d_{i,j} \neq 0\}$. We then measure the similarity of $\mathbf{d}_i$ to $\mathbf{d}_j$ by the following expression:

$$w(i\|j) = \sum_{k \in \mathcal{X}_i} \frac{\tilde{d}_{i,k} \tilde{d}_{j,k}}{\sqrt{\sum_{k \in \mathcal{X}_i} \tilde{d}_{i,k}^2} \sqrt{\sum_{k \in \mathcal{X}_i} \tilde{d}_{j,k}^2}} \tag{5.7}$$

This actually makes intuitive sense: a short document may have large similarity to a long

document with diverse topics while the long document may not have the large similarity to the short one because of its diverse topics.

5.3 Case Study I: Binary Classification

We evaluate our proposed approach with binary text classification tasks. Generally, the following questions will be addressed:

- *Is the classification performance improved by using directed graph?* We compare label propagation approaches using directed graphs with approaches using undirected graphs on the same classification tasks. SGT [23], SVM [21] and LP [57] are used as our baselines.

- *Why do directed graphs improve the performance?* From the experiments we can see that propagation using directed graphs outperforms using undirected graphs. But how it actually helps classification and why this is the case still need to be studied. We'll discuss why directed graphs actually help classification with the analysis of the dataset.

- *How sensitive are two proposed asymmetric similarity measures?* We proposed two similarity measures above. How they are impacted by different parameters will be presented and analyzed.

5.3.1 Experiment Setup

We evaluate the proposed algorithm for semi-supervised text categorization by multiple datasets which are described as follows:

- *Movielens dataset*: We use the *MovieLens* dataset[1]. The five most popular categories are used in this study, including Adventure, Children's, Comedy, Drama and Thriller,

[1] `http://www.grouplens.org/`

which results in a total of 1354 movies. Each movie is described by a number of keywords, and the goal of this experiment is to predict the genre of movies based on their keyword description.

- *20-newsgroup*: For *20-newsgroup* dataset, we randomly selected 5 categories and 100 documents from each category. The resulting dataset contains 500 documents and around 7300 words. There is no overlapping of documents among categories. That is to say, a document is assigned to only one category.

- *Reuters-21578*: *Reuters-21578* dataset contains more than 20,000 documents. We used the subset *Reuters-21578 R8* dataset[2] which contains 7674 documents and 8 classes. Each document is assigned to only one category. Since some classes don't have enough samples, we select 6 classes which have more than 80 documents. The resulting dataset has 543 documents with the vocabulary of 4452 words and the average number of documents within one class is around 91.

Three baselines are used in our study: the supervised support vector machine (SVM), the spectral graph transducer (SGT) over undirected graphs where document similarity is computed with cosine similarity and the label propagation using harmonic function (LP) in [57]. SVM is a supervised learner which only utilizes the labeled data. SGT explores the undirected graph with an adjacency matrix based on k-nearest neighbors and propagates the labels over the undirected graph. LP propagates the labels over an undirected graph using gaussian fields and harmonic function.

We refer to the directed graph label propagation based on the asymmetric cosine similarity and the KL divergence as DP-Con and DP-KL, respectively. $F1$ is used as the evaluation metric. All the experiments are repeated ten times, and $F1$ averaged over ten trials is reported as the final result.

[2] http://www.gia.ist.utl.pt/~acardoso/datasets/

5.3.2 Experiment Results

Table 5.3.2, 5.2 and 5.3 present the F1 scores of five algorithms in different binary classification tasks for three datasets. First, as we can see from three tables, DP-KL and DP-Con always outperform SGT and LP which are both semi-supervised learning algorithms using undirected graphs. We attribute the improvement to utilizing the directed graphs during propagation since the directed graph can better capture the asymmetric relationship between documents in many cases. Detailed discussion about why it is the case will be presented later.

Second, generally speaking, except LP algorithm, semi-supervised learning algorithms (DP-KL, DP-Con and SGT) perform better than SVM which is a supervised learning algorithm. This is reasonable because the deficiency of training examples will affect the effectiveness of supervised learning. This is again proved that utilizing unlabeled data is important for classification tasks especially when a large amount of training data is not available. As the size of training set increases, the performance of SVM is usually improved significantly since more training samples can be used for learning. The performance of SVM varies from case to case and in some cases shown (for example, the category "Childre's" in Table 5.3.2, the category "trade" in Table 5.3), it outperforms SGT, but not DP-KL and DP-Con. This supports out estimation that directed graphs can improve the performance.

Third, LP performs poorly in all cases. We attribute this to two possible reasons. First, the poor performance may be caused by propagating the labels over the weighted undirected graph based on the similarity matrix in which each pair of nodes are connected and the edge between them is assigned a weight. The labeling information from connected edges with small weights may become very noisy during propagation. Thus it improves the performance by using adjacency matrix with k nearest-neighbor which removes unnecessary edges from the graph. Another reason may be the threshold for classification. The prediction result from harmonic function seems very sensitive to the threshold. Considering the class prior is one reasonable way to adjust the threshold. It works usually better

than simply taking the sign of the final value. But compared to other algorithms, LP doesn't seem always work well.

		Number of Training examples			
Cat.	Alg.	10	20	40	80
Adv.	DP-KL	0.201 (0.044)	0.209 (0.042)	0.210 (0.034)	0.220 (0.004)
	DP-Con	0.225 (0.055)	0.244 (0.053)	0.252 (0.050)	0.264 (0.047)
	SGT	0.149 (0.026)	0.183 (0.032)	0.2036 (0.034)	0.208 (0.044)
	SVM	0.111 (0.011)	0.170 (0.067)	0.229 (0.058)	0.346 (0.010)
	LP	0.155 (0.002)	0.157 (0.013)	0.156 (0.008)	0.163 (0.005)
Chi.	DP-KL	0.194 (0.046)	0.197 (0.040)	0.196 (0.052)	0.198 (0.017)
	DP-Con	0.161 (0.068)	0.2069 (0.087)	0.264 (0.091)	0.281 (0.068)
	SGT	0.139 (0.038)	0.147 (0.022)	0.156 (0.025)	0.159 (0.036)
	SVM	0.137 (0.085)	0.155 (0.020)	0.180 (0.056)	0.189 (0.016)
	LP	0.071 (0.003)	0.074 (0.021)	0.072 (0.022)	0.080 (0.005)
Com.	DP-KL	0.412 (0.069)	0.422 (0.043)	0.440 (0.038)	0.444 (0.034)
	DP-Con	0.381 (0.042)	0.404 (0.035)	0.411 (0.049)	0.421 (0.038)
	SGT	0.373 (0.0676)	0.395 (0.057)	0.405 (0.047)	0.413 (0.025)
	SVM	0.311 (0.111)	0.321 (0.053)	0.345 (0.074)	0.363 (0.065)
	LP	0.296 (0.223)	0.294 (0.002)	0.289 (0.004)	0.293 (0.006)
Dra.	DP-KL	0.630 (0.063)	0.639 (0.053))	0.678 (0.044)	0.702 (0.020)
	DP-Con	0.634 (0.024)	0.659 (0.051)	0.6612 (0.075)	0.700 (0.017)
	SGT	0.583 (0.071)	0.588 (0.105)	0.647 (0.039)	0.678 (0.031)
	SVM	0.348 (0.074)	0.3979 (0.054)	0.416 (0.014)	0.546 (0.055)
	LP	0.353 (0.002)	0.3601 (0.002)	0.427 (0.004)	0.431 (0.005)
Thri.	DP-KL	0.1985 (0.047)	0.201 (0.034)	0.209 (0.037)	0.213 (0.023)
	DP-Con	0.220 (0.044)	0.230 (0.044)	0.231 (0.046)	0.245 (0.029)
	SGT	0.172 (0.028)	0.181 (0.034)	0.185 (0.028)	0.195 (0.020)
	SVM	0.128 (0.017)	0.131 (0.058)	0.153 (0.060)	0.165 (0.027)
	LP	0.136 (0.003)	0.139 (0.002)	0.141 (0.006)	0.161 (0.007)

Table 5.1: The F1 results for *MovieLens* Dataset.

		Number of Training examples			
Cat.	Alg.	10	20	30	40
1	DP-KL	0.585 (0.020)	0.606 (0.069)	0.611 (0.057)	0.623 (0.076)
	DP-Con	0.616 (0.042)	0.628 (0.042)	0.635 (0.043)	0.653 (0.076)
	SGT	0.570 (0.075)	0.578 (0.071)	0.595 (0.070)	0.604 (0.080)
	SVM	0.332 (0.055)	0.434 (0.070)	0.469 (0.022)	0.594 (0.008)
	LP	0.245 (0.007)	0.261 (0.013)	0.285 (0.011)	0.304 (0.016)
2	DP-KL	0.557 (0.044)	0.562 (0.039)	0.572 (0.047)	0.589 (0.042)
	DP-Con	0.629 (0.041)	0.652 (0.055)	0.663 (0.052)	0.667 (0.047)
	SGT	0.544 (0.045)	0.554 (0.024)	0.567 (0.066)	0.572 (0.065)
	SVM	0.399 (0.053)	0.422 (0.044)	0.514 (0.024)	0.592 (0.013)
	LP	0.361 (0.007)	0.376 (0.009)	0.385 (0.012)	0.395 (0.021)
3	DP-KL	0.562 (0.052)	0.594 (0.044)	0.629 (0.057)	0.626 (0.070)
	DP-Con	0.594 (0.054)	0.614 (0.035)	0.642 (0.051)	0.641 (0.051)
	SGT	0.544 (0.040)	0.559 (0.064)	0.566 (0.065)	0.570 (0.084)
	SVM	0.365 (0.055)	0.520 (0.053)	0.573 (0.074)	0.607 (0.069)
	LP	0.249 (0.008)	0.271 (0.013)	0.293 (0.015)	0.317 (0.015)
4	DP-KL	0.567 (0.046)	0.571 (0.066)	0.596 (0.064)	0.607 (0.060)
	DP-Con	0.614 (0.027)	0.632 (0.035)	0.639 (0.051)	0.640 (0.074)
	SGT	0.542 (0.031)	0.561 (0.058)	0.567 (0.068)	0.590 (0.090)
	SVM	0.382 (0.055)	0.460 (0.027)	0.515 (0.069)	0.600 (0.013)
	LP	0.382 (0.004)	0.411 (0.007)	0.420 (0.016)	0.427 (0.018)
5	DP-KL	0.477 (0.053)	0.487 (0.035)	0.500 (0.045)	0.501 (0.047)
	DP-Con	0.599 (0.038)	0.610 (0.041)	0.616 (0.043)	0.630 (0.033)
	SGT	0.459 (0.069)	0.462 (0.037)	0.462 (0.046)	0.473 (0.059)
	SVM	0.253 (0.072)	0.308 (0.009)	0.451 (0.011)	0.533 (0.069)
	LP	0.154 (0.008)	0.168 (0.010)	0.187 (0.013)	0.206 (0.020)

Table 5.2: F1 measure for *20-newsgroup* Dataset.

		Number of Training examples			
Cat.	Alg.	10	20	30	40
acq	DP-KL	0.514 (0.080)	0.535(0.083)	0.560 (0.097)	0.591 (0.086)
	DP-Con	0.527(0.050)	0.527 (0.051)	0.548 (0.041)	0.569 (0.048)
	SGT	0.504 (0.050)	0.531 (0.054)	0.541 (0.045)	0.549 (0.036)
	SVM	0.362 (0.029)	0.446 (0.010)	0.527 (0.049)	0.535 (0.042)
	LP	0.177 (0.005)	0.177 (0.010)	0.181 (0.013)	0.188 (0.013)
crude	DP-KL	0.773 (0.056)	0.811 (0.090)	0.813 (0.021)	0.821 (0.015)
	DP-Con	0.549 (0.041)	0.572 (0.045)	0.573 (0.051)	0.583 (0.054)
	SGT	0.540 (0.033)	0.561(0.051)	0.589 (0.057)	0.579 (0.058)
	SVM	0.482 (0.084)	0.529(0.037)	0.550 (0.046)	0.563 (0.101)
	LP	0.294 (0.009)	0.304(0.011)	0.310 (0.015)	0.315 (0.017)
earn	DP-KL	0.696(0.112)	0.731 (0.089)	0.705 (0.102)	0.789 (0.072)
	DP-Con	0.595 (0.038)	0.610 (0.044)	0.635 (0.043)	0.647 (0.068)
	SGT	0.572 (0.029)	0.595 (0.031)	0.606 (0.051)	0.614 (0.063)
	SVM	0.508 (0.092)	0.571 (0.120)	0.600 (0.140)	0.608 (0.026)
	LP	0.223 (0.018)	0.242 (0.016)	0.271 (0.030)	0.311 (0.026)
interest	DP-KL	0.470 (0.052)	0.505 (0.059)	0.483 (0.065)	0.537 (0.075)
	DP-Con	0.447 (0.047)	0.492 (0.061)	0.478 (0.044)	0.485 (0.052)
	SGT	0.439 (0.054)	0.490 (0.052)	0.464 (0.044)	0.480 (0.061)
	SVM	0.403 (0.036)	0.472 (0.032)	0.489 (0.022)	0.434 (0.067)
	LP	0.228 (0.012)	0.230 (0.011)	0.230 (0.017)	0.233 (0.014)
money	DP-KL	0.471 (0.074)	0.528 (0.071)	0.556 (0.080)	0.574 (0.078)
	DP-Con	0.471 (0.066)	0.498 (0.062)	0.506 (0.071)	0.506 (0.050)
	SGT	0.456 (0.063)	0.490 (0.061)	0.484 (0.060)	0.490 (0.042)
	SVM	0.382 (0.018)	0.436 (0.013)	0.458 (0.021)	0.487 (0.022)
	LP	0.208 (0.007)	0.211 (0.012)	0.212 (0.012)	0.212 (0.012)
trade	DP-KL	0.505 (0.090)	0.550 (0.106)	0.576 (0.041))	0.608 (0.057)
	DP-Con	0.494 (0.054)	0.510 (0.058)	0.567 (0.043)	0.582 (0.054)
	SGT	0.463 (0.057)	0.462 (0.061)	0.462 (0.054)	0.467 (0.062)
	SVM	0.478 (0.152)	0.492 (0.095)	0.506 (0.070)	0.523 (0.077)
	LP	0.347 (0.007)	0.351 (0.009)	0.354 (0.012)	0.318 (0.012)

Table 5.3: F1 measure for *Reuters-21578 R8* Dataset.

5.3.3 Analysis and Discussion

The results presented above show that propagation using directed graphs works better than propagation using undirected graphs. However, it may not always be the case. In this section, we give a closer look at why directed graphs can bring the improvement of performance. We consider SGT, DP-KL and DP-Con.

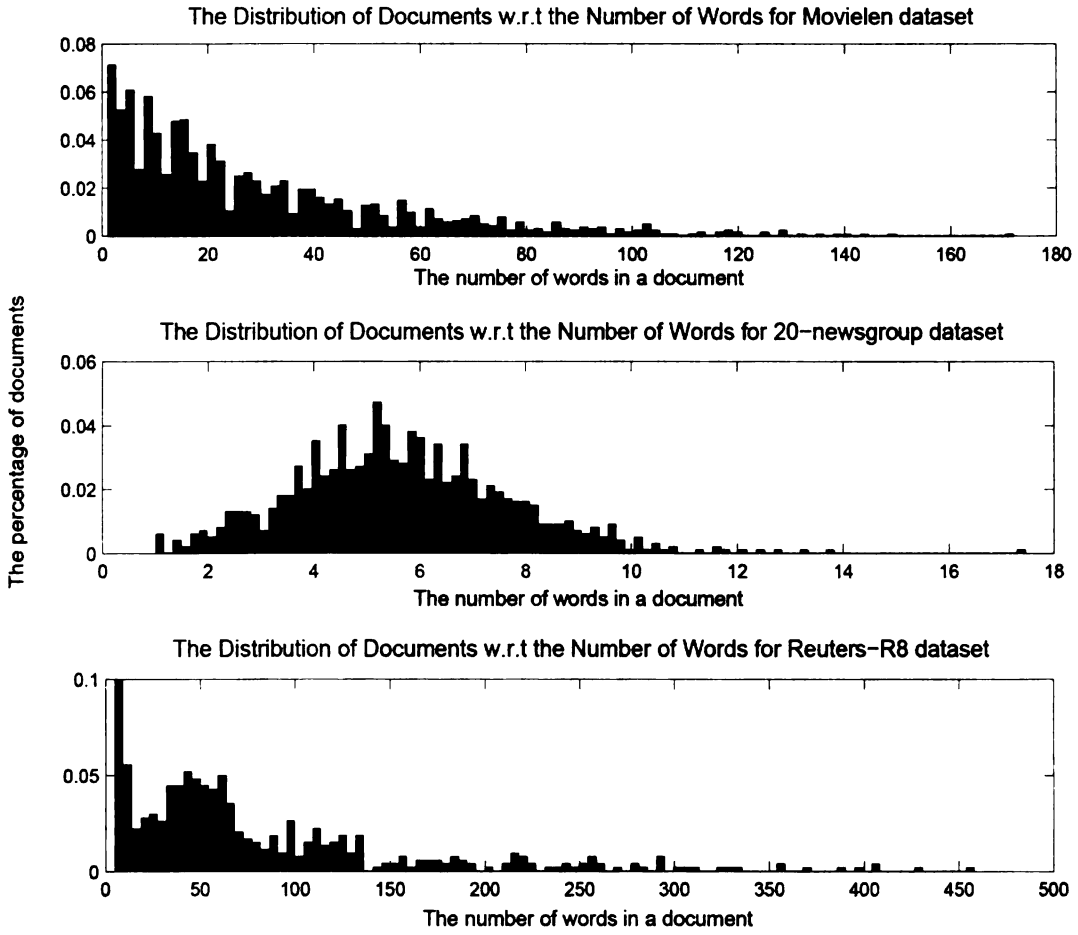
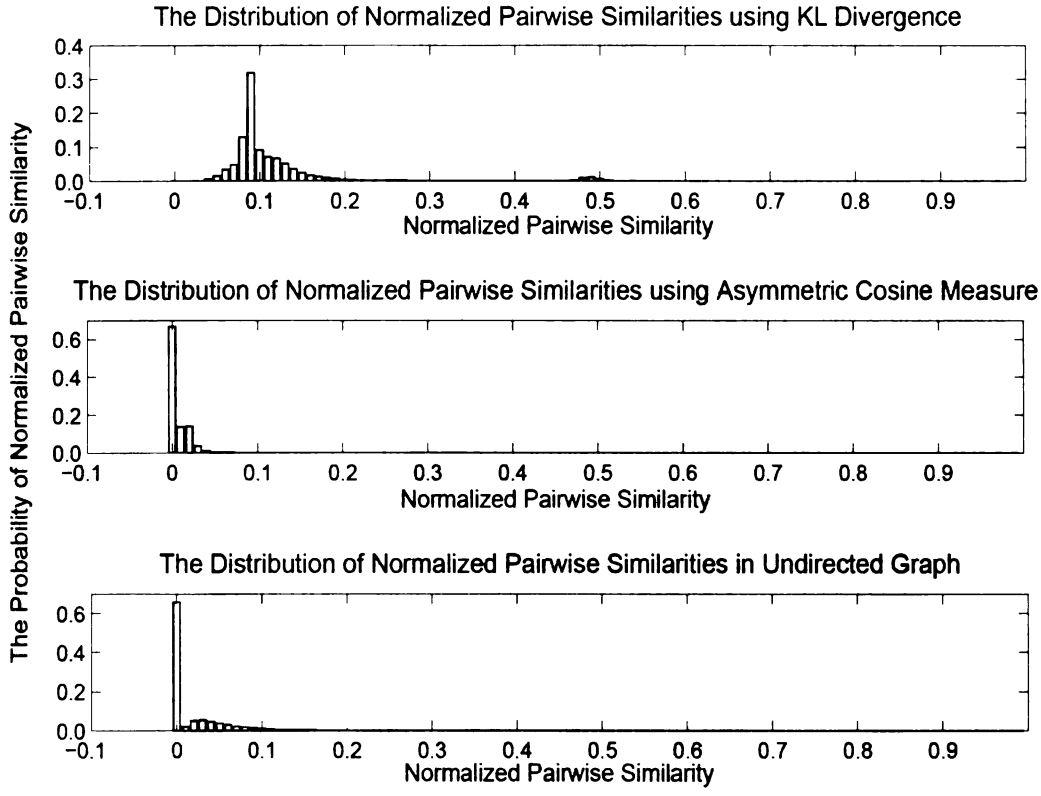
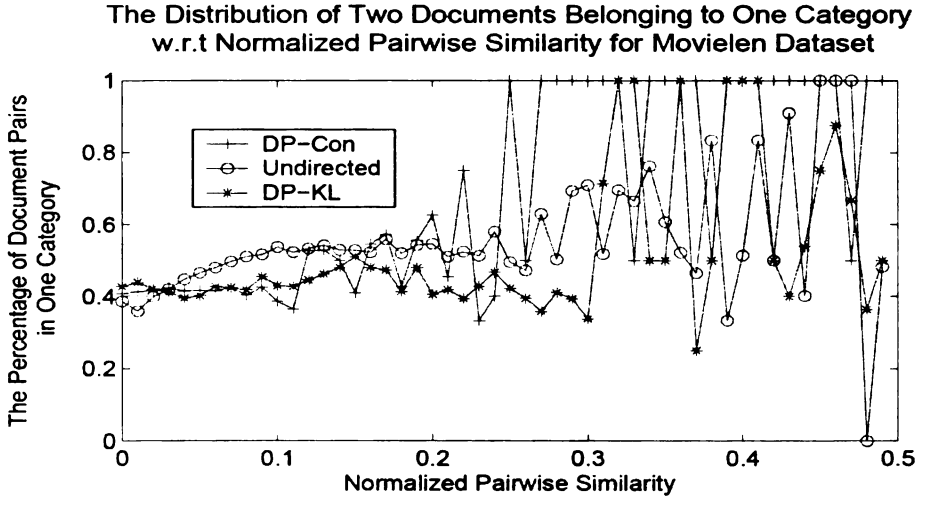


Figure 5.2: The Distribution of Document Length in Three Datasets.

Why does propagation using directed graphs outperform undirected graph? We already show that propagation using directed graphs outperforms propagation using undirected graphs in text categorization task. Recall that we use SGT algorithm for propagation with the undirected graph converted from directed graph. The performance of each algo-

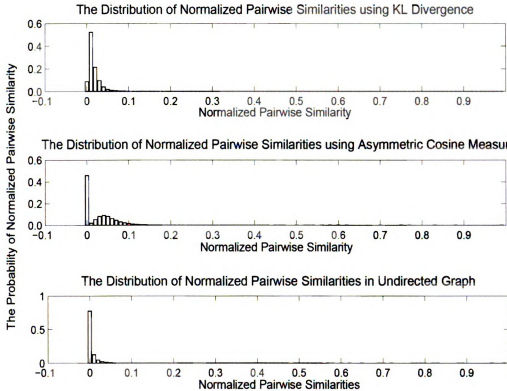


(a) The Distribution of Pairwise Similarities in *MovieLens* Dataset.

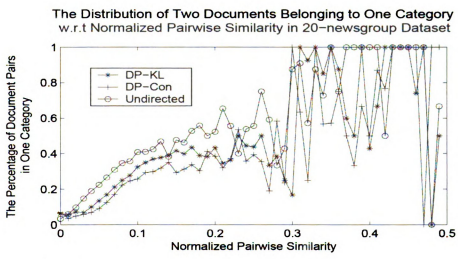


(b) The Distribution of Two documents Belonging to One Category with respect to Normalized Pairwise Similarity in *MovieLens* Dtaset.

Figure 5.3: Similarity Matrix Analysis of *MovieLens* Dataset

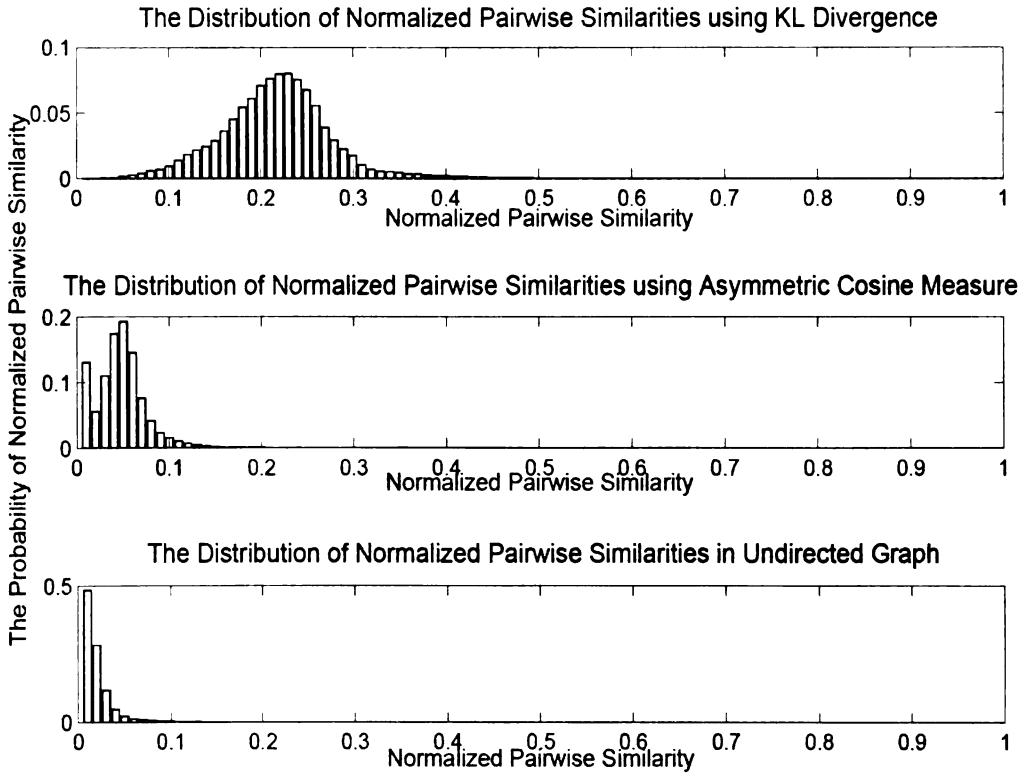


(a) The Distribution of Pairwise Similarities in *20-newsgroup* Dataset.

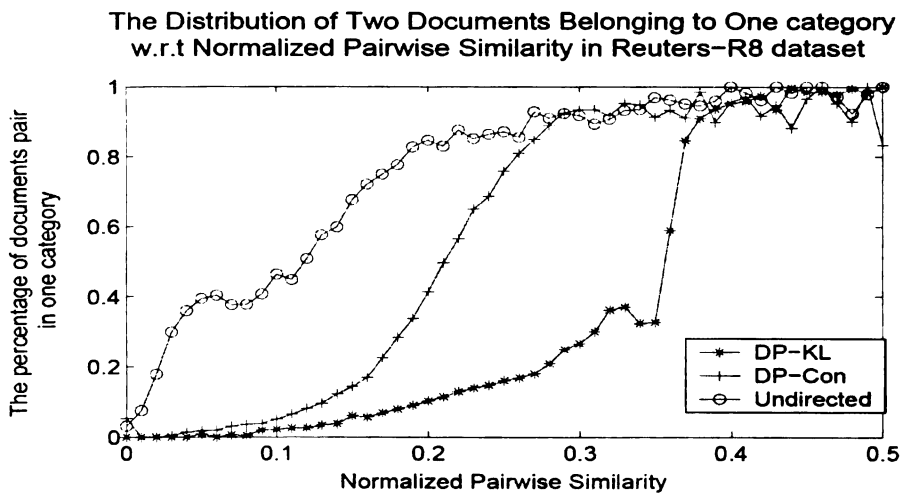


(b) The Distribution of Two Documents Belonging to One Category with respect to Normalized Pairwise Similarity in *20-newsgroup* Dataset.

Figure 5.4: Similarity Mtrix Analysis of 20-newsgroup Dataset



(a) The Distribution of Pairwise Similarities in *Reuters-21578 R8* Dataset.



(b) The Distribution of Two Documents Belonging to One Category with respect to Normalized Pairwise Similarity in *Reuters-21578 R8* Dataset.

Figure 5.5: Similarity Matrix Analysis of *Reuters-21578 R8* Dataset

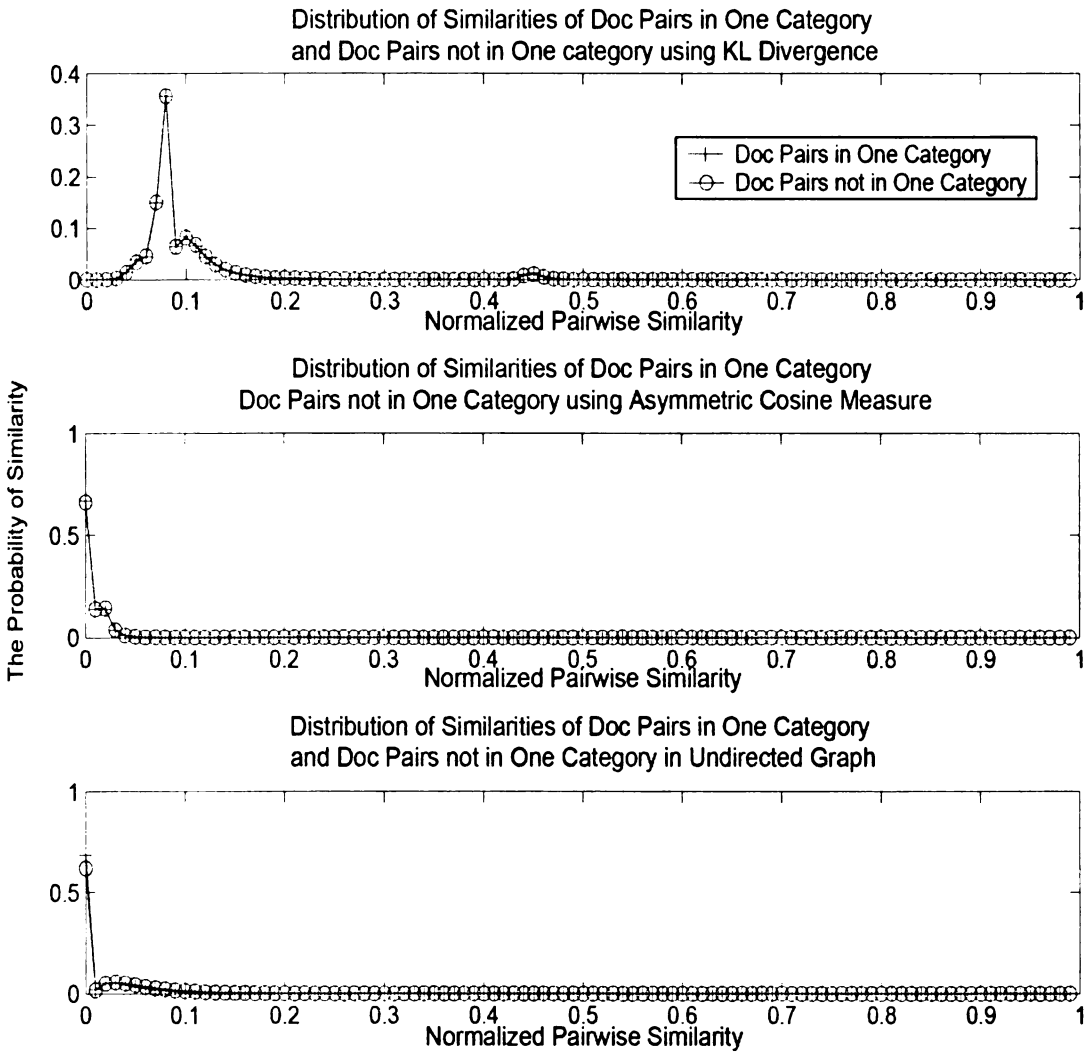


Figure 5.6: The Distribution of Similarities of Document Pairs Belonging to the Same Category and Document Pairs not Belonging to the Same Category in *MovieLens* Dataset.

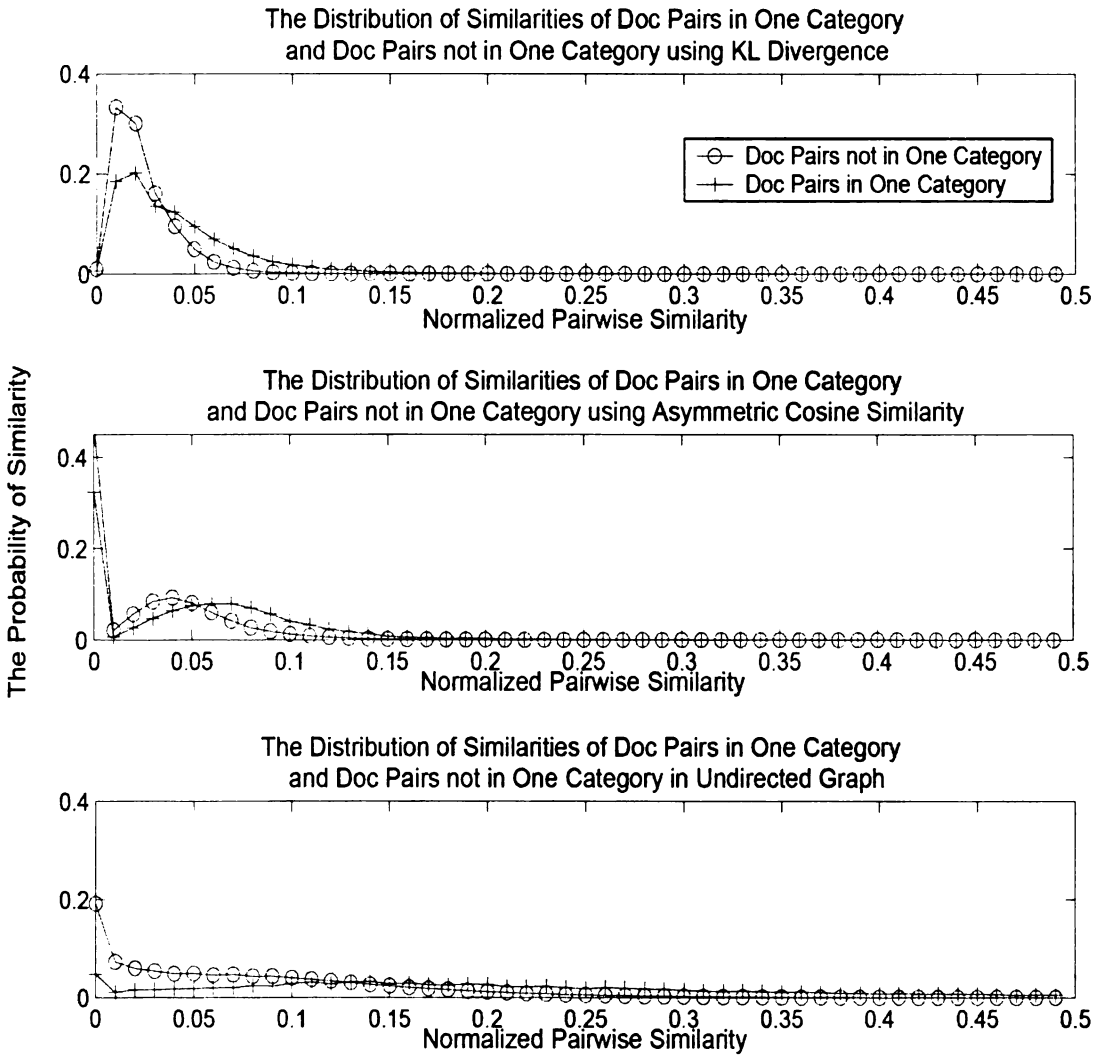


Figure 5.7: The Distribution of Similarities of Document Pairs Belonging to the Same Category and Document Pairs not Belonging to the Same Category in *20-newsgroup* dataset.

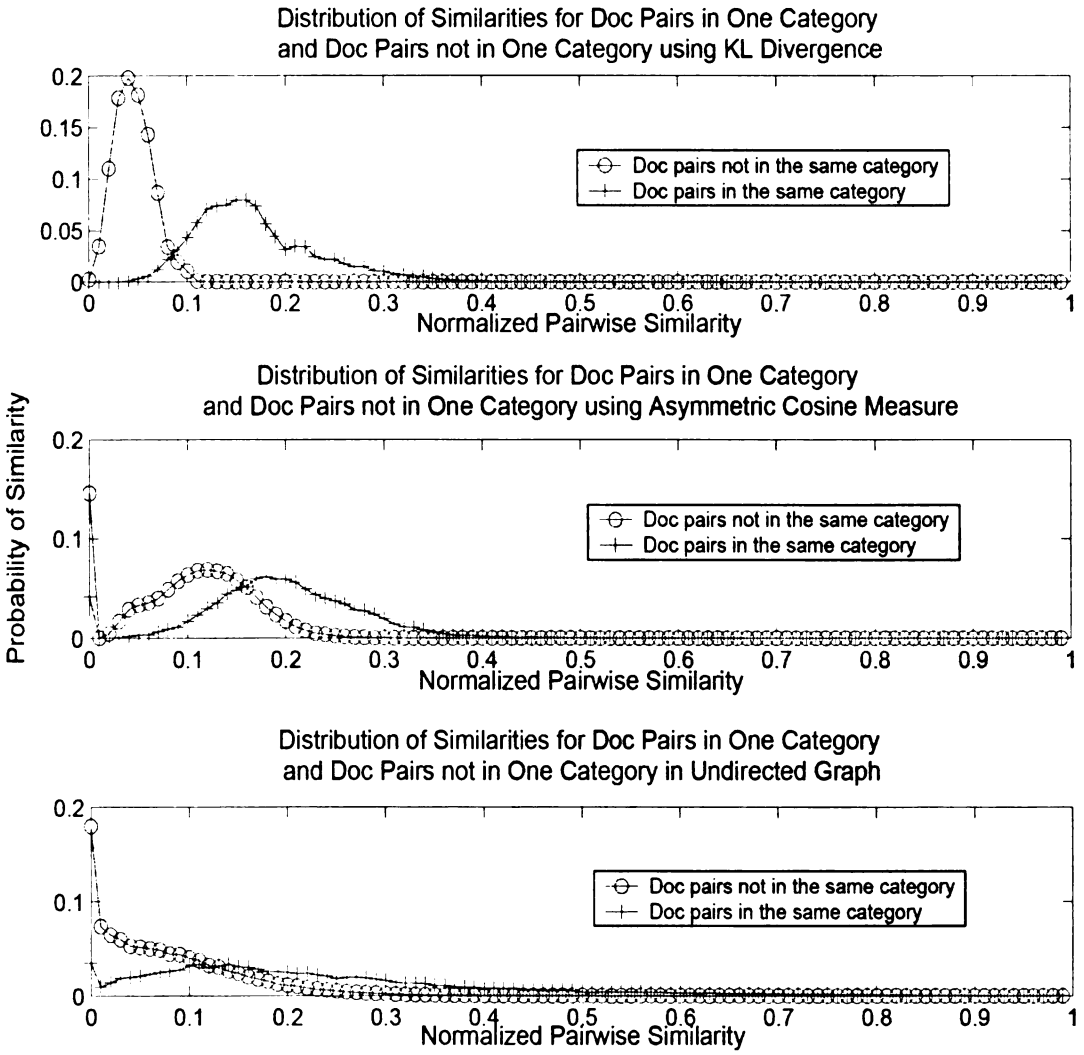


Figure 5.8: The Distribution of Similarities of Document Pairs Belonging to the Same Category and Document Pairs not Belonging to the Same Category in *Reuters-21578 R8* dataset.

rithm simply depends on the similarity matrix. Thus, in order to understand the reason behind it, we study the similarity matrix computed in DP-KL, DP-Con and SGT. DP-KL and DP-Con generate their similarity matrices based on KL divergence and asymmetric cosine similarity respectively. For SGT, the similarity between two documents is computed by using standard cosine similarity (symmetric).

First, we claim that the correlation between pairwise similarity and the probability of two document belonging to the same category is enhanced in directed graphs using KL divergence based similarity and asymmetric cosine similarity. Figure 5.3(b), 5.4(b) and 5.5(b) present the distribution of two documents belonging to the same category with respect to the normalized pairwise similarity in *MovieLens* dataset, *20-newsgroup* dataset and *Reuters-21578 R8* dataset. In three figures, the probability of two documents belonging to the same category increases significantly at some high similarity point for directed graph while in undirected graph, the probability does not have a sharp change and instead increases gradually as the similarity get larger. This shows that it is more likely for KL divergence based similarity and asymmetric cosine similarity to assign higher similarity values to document pairs belongs to the same category.

To further understand this point, we plot the similarity distribution of all document pairs belonging to the same category and all document pairs not belonging to the same category for three datasets in 5.6, 5.7 and 5.8. For *Reuters-21578 R8* dataset (Figure 5.8), two kinds of similarities are clearly distributed in two modes for directed graphs. That is, the document pairs belonging to the same category are more likely to have higher pairwise similarities and document pairs not belonging to the same category are more likely to have lower pairwise similarities. For *20-newsgroup* (Figure 5.7), this trend can be seen although not as clear as in *Reuters-21578 R8* dataset. This may be the reason why propagation using directed graphs significantly outperforms using undirected graphs in *Reuters-21578 R8* dataset while not in *20-newsgroup* dataset. We also plot the same distribution for *MovieLens* dataset in Figure 5.6, but two distributions are not separated very obviously. This may

explain why the performance improvement from directed graphs for *MovieLens* dataset is not very significant. Our assumptions for using directed graphs are based on the analysis of document length. These assumptions may not work for all data sets because of the representation of the data, accuracy and some other reasons. Accordingly the similarity measures based on the assumptions will not be able to accurately express the relationship among data points. Thus the performance gained from using directed graphs was not that obvious compared to undirected graphs.

From the above analysis, we gain some insight into why propagation using directed graphs can outperform propagation using undirected graphs in some scenarios. Directed graphs based two proposed asymmetric measure can better explore the manifold structure of the dataset and reflect the impact of document length, term frequency distribution and other factors on classification in some cases as in *Reuter-R8* dataset and *20-newsgroup* dataset. Thus the performance was improved by using directed graphs. However, in *MovieLens* dataset, since the directed graphs didn't capture the appropriate properties of the dataset, the performance was not significantly improved though this may be solved by using some other different similarity measure. In summary, directed graphs constructed from plausible assumptions will definitely improve the performance of label propagation and are worthwhile studying.

The sensitivity of asymmetric similarity measures We proposed two asymmetric similarity measures, namely *KL divergence similarity* and *asymmetric cosine similarity*. Since there is no parameter tuning in *asymmetric cosine similarity*, we leave it out of our discussion. *KL divergence similarity* uses RBF kernel ($w(i\|j) = \exp(-\lambda D(i\|j))$, see section 5.2) to convert the distance into similarity and a scaling factor λ is involved in the process. We briefly discuss how the parameter λ affects the performance.

Table 5.4 shows the F1 scores of DP-KL on *20-newsgroup* dataset with different λ values. $\lambda = 10$ has the best performance. The result for $\lambda = 5$ is much worse than $\lambda = 10$.

For the values greater than 10, as λ is increasing , the performance is decreasing in most cases. This is consistent with the meaning of λ which is the scaling factor of distance. Too small or too big scaling values will both result in the inaccurate similarity.

		Number of Training examples			
Cat.	λ =	10	20	30	40
1	5	0.3971 (0.0613)	0.4431 (0.0813)	0.4996 (0.0662)	0.5055 (0.0641)
	10	0.5845 (0.0197)	0.6059 (0.0688)	0.6108 (0.0570)	0.6226 (0.0762)
	20	0.4221 (0.0773)	0.4436 (0.0809)	0.4781 (0.0688)	0.5178 (0.0577)
	30	0.4097 (0.0952)	0.4568 (0.0817)	0.4849 (0.0612)	0.5001 (0.0837)
	50	0.3807 (0.0848)	0.4295 (0.0710)	0.4870 (0.0730)	0.5102 (0.0708)
2	5	0.4371 (0.1013)	0.4695 (0.0987)	0.5253 (0.0895)	0.4719 (0.0831)
	10	0.6287 (0.0414)	0.6515 (0.0551)	0.6631 (0.0517)	0.6636 (0.0471)
	20	0.5391 (0.0089)	0.5862 (0.0872)	0.5868 (0.0854)	0.5937 (0.0773)
	30	0.4433 (0.1296)	0.4378 (0.0731)	0.4777 (0.0981)	0.4947 (0.0804)
	50	0.4040 (0.1232)	0.4759 (0.1137)	0.4955 (0.0936)	0.5283 (0.0880)
3	5	0.4349 (0.0469)	0.5364 (0.1061)	0.5659 (0.0855)	0.5709 (0.0892)
	10	0.5620 (0.0523)	0.5939 (0.0440)	0.6290 (0.0574)	0.6257 (0.0691)
	20	0.5049 (0.0305)	0.5257 (0.1024)	0.5571 (0.0869)	0.5757 (0.0772)
	30	0.3879 (0.1210)	0.5326 (0.0836)	0.5570 (0.0885)	0.6191 (0.0656)
	50	0.4984 (0.0504)	0.5231 (0.0901)	0.5588 (0.0721)	0.5731 (0.0795)
4	5	0.4159 (0.0782)	0.4667 (0.0761)	0.5134 (0.0719)	0.5052 (0.0733)
	10	0.5674 (0.0458)	0.5709 (0.0656)	0.5954 (0.0643)	0.6072 (0.0601)
	20	0.4019 (0.0975)	0.4918 (0.0838)	0.5127 (0.0698)	0.5214 (0.0762)
	30	0.3749 (0.0756)	0.4574 (0.0831)	0.5001 (0.0791)	0.5118 (0.0732)
	50	0.4086 (0.0689)	0.4758 (0.0761)	0.4954 (0.0643)	0.5062 (0.0709)
5	5	0.3491 (0.0475)	0.3532 (0.0530)	0.3830 (0.0587)	0.4049 (0.0540)
	10	0.4766 (0.0531)	0.4866 (0.0350)	0.4981 (0.0454)	0.5006 (0.0473)
	20	0.4274 (0.0665)	0.4563 (0.0763)	0.4812 (0.0407)	0.4815 (0.0496)
	30	0.3130 (0.0687)	0.3696 (0.0645)	0.3877 (0.0445)	0.4001 (0.0572)
	50	0.3250 (0.0629)	0.3556 (0.0600)	0.3908 (0.0389)	0.3941 (0.0608)

Table 5.4: The $F1$ Results for *20-newsgroup* Dataset with Different λ Values.

5.4 Case Study II: Multi-label Classification

We extent the idea of Propagation over directed graphs to the multi-label classification task on the *MovieLens* dataset. The goal of the experiments is to give a more comprehensive view of the effectiveness of propagation over directed graphs.

We conduct the experiment of propagation using KL divergence based similarity measure as discussed above. We follow the propagation scheme proposed in [53]. For each category, the propagation will generate the final score for each document. We view those scores as confidence scores of one document belonging to one category. Thus, by ranking the scores of all categories for each document, we can make predictions accordingly.

5.4.1 Experiment Setup

We again use *MovieLens* dataset[3] as our testbed. The dataset provides the movie category information. There are originally 1682 movies and 19 categories. The average number of movies for each category is around 152 and the average number of categories for each movie is 2.

We also downloaded the movie keyword information. After we removed the movies with no keywords, the resulting movie keyword file contains 1651 movies with 11107 keywords. We remove the corresponding movies from the movie category information file and the resulting movie category file also contains the same 1651 movies. The average number of movies per category is around 1 and the average number of categories per movie is around 4.

We refer to the propagation approach on the directed graph as *Directed Propagation* (DP) and compare it with label propagation approach in [53]. We refer to the label propagation approach as LP. According to the figure 5.1, directed propagation needs the input $S^{\mathcal{D}}$ which is the asymmetric movie similarity matrix. We compute the $S^{\mathcal{D}}$ from movie key-

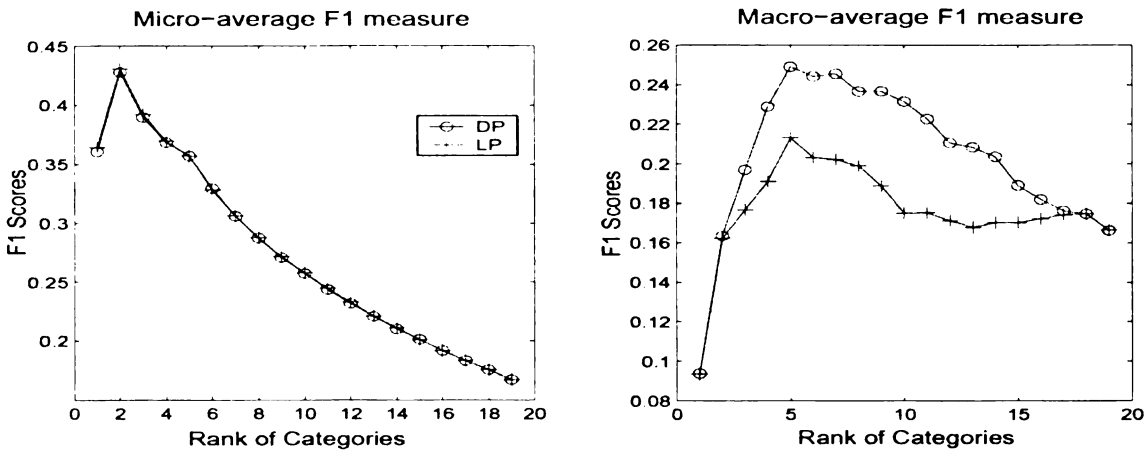
[3] http://www.grouplens.org/

words information using K-L divergence in Equation (5.6). LP algorithm has also the input $S^{\mathcal{D}}$ which is the symmetric movie similarity matrix and we compute it from movie ratings information using cosine similarity.

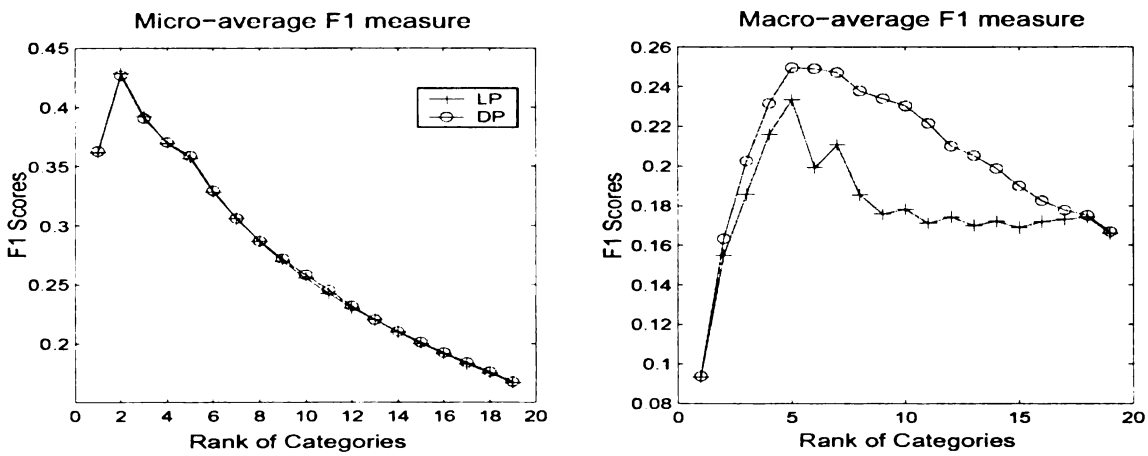
We use F1 measure as our evaluation metrics since it is a more appropriate measure for evaluating both precision and recall. We refer to the average F1 measure per category as "macro-average" F1 measure and the average F1 measure per movie as "micro-average" F1 measure. Since both DP and LP algorithms provide a ranking of categories for each movie, we compute the F1 measure at different ranks. For macro-average F1 measure, precision and recall for each category are computed; the average precision and recall across all categories are taken for computing macro-average F1 measure. For micro-average F1 measure, precision and recall for each movie are computed; the average across all movies are taken for computing micro-average F1 measure. All experiments are conducted 10 times and the average across 10 trials are used as the final results.

5.4.2 Results and Analysis

Figure 5.9(a) and 5.9(b) shows the performance of DP and LP algorithms for 40% and 20% training data respectively. For micro-average F1 measure, two algorithms performs almost the same. For macro-average F1 measure, DP algorithm performs significantly better than LP algorithm. This may be explained by the way of constructing the directed graph. Since we compute the asymmetric weight matrix from movie keyword information, the movies from the same categories may have larger similarity because they share more keywords. More in-depth research has to be done in order to have a better understanding. We leave it to the future work.



(a) 40% training data.



(b) 20% training data.

Figure 5.9: F1 Measure at Different Ranks for *MovieLens* Dataset in Multi-label Classification Task.

Chapter 6

Conclusion and Future Work

This dissertation presents a comprehensive study of label propagation with its applications. The related research works of label propagation has been discussed in detail. Several propagation schemes, namely, *relation propagation*, *rank propagation* and *propagation on directed graph* are proposed in order to solve the existing problems in label propagation. The applications of the proposed schemes are presented with the empirical studies, which has shown that the proposed label propagation methods can achieve better performance than standard label propagation approaches in many problems. This chapter will start with summarizing the proposed work in this thesis followed by the discussion of some interesting directions in future work.

6.1 Summary

Label propagation is an effective approach to semi-supervised learning. The main idea behind label propagation is to first construct a graph by denoting each data example as a node and connecting each pairs of nodes with an edge assigned with a weight (usually similarity between corresponding data examples), then propagate the labels of known data examples and finally make the prediction according to the propagation scores. Empirical study has shown it's an effective approach in a lot of applications. Although label propagation is a

very promising approach, it still has limitations and disadvantages in many scenarios. We discussed some of its limitations and proposed several new propagation schemes accordingly.

Relation Propagation We proposed the generalized framework of *Relation Propagation* and discussed its applications including multi-label classification task and collaborative filtering task. In most previous work in label propagation, propagation is conducted among one single type of objects. However, real applications usually involve multiple types of objects. Propagation over only one type of objects will usually miss the correlations among different types of objects and we believe that the correlations among multiple types of objects will help in the propagation process. Based on this assumption, we propose the framework of relation propagation which propagates the *relations* among multiple types of objects instead of propagating the *labels* directly among one type of objects.

Consider the case of two types of objects. We construct the graph in which each node in the graph represents a object pair with one object from each type. Each edge connecting two nodes in the graph is assigned with a user-defined weight. We proposed to use the direct product of two similarity matrices respectively from two types of objects. The relations among two types of objects should be defined according to different situations. For examples, in multi-label classification task, two types of objects are documents and categories and the relation is the membership of documents in categories. Then by propagating the relations over the constructed graph, we achieve for each unlabeled example the confidence scores of this example belonging to all categories. The empirical study showed that the relation propagation is a more effective approach in some cases and it proved our assumption that it is helpful to utilize the correlations among multiple types of objects.

Rank Propagation It is necessary to study the label propagation approaches on ranked data due to the limitations of label propagation. First, most label propagation models propagate directly the class labels. The problem may arise because the ordering information

among the numerical values representing the class labels is incorrectly introduced to the propagation process. One way to avoid this issue is to cast the problem as a ranking problem in which the label information is converted to pairwise preferences over classes for each training example and these preferences are propagated. Second, most previous research works on label propagation require the labels of known data for training. However, the label information is not always available. Instead, in a lot of ranking applications including web page searching and collaborative filtering, usually only the ordering information is provided. Previous label propagation models are often not appropriate in these cases.

To address the challenges, a *Rank Propagation* scheme for multi-label categorization scenario is proposed. Instead of propagating the *labels* of training examples, rank propagation actually propagates the category *preference* information from all labeled data. Specifically, given the labels of all training examples, a preference matrix over all category pairs is built for each training example. Then, for each testing example, an optimization problem is constructed which minimizes the weighted differences between the preference matrix to be computed for the testing example and the preference matrix of each training example. As a result, for each unlabeled example, a preference matrix is computed and a ranking list of all categories is generated by computing the principle eigenvector of the preference matrix. Two loss functions, namely trace-based loss function and a more general loss function, are designed for measuring the difference between two preference matrices. Empirical study with multiple datasets has shown that rank propagation scheme is an effective approach.

Propagation over Directed Graphs Most previous research on label propagation was conducted on undirected graphs. For propagation over undirected graphs, the similarity matrix is symmetric. But in many cases, relationship among objects can be asymmetric and is more appropriate to be expressed by directed graphs which result in asymmetric matrices. For instance, web pages connected with hyperlinks are better presented with a

directed graph.

The framework of *Propagation over Directed Graphs* is proposed to utilize the directed graphs. In order to construct directed graphs, two asymmetric similarity measures are designed: KL divergence-based similarity and Asymmetric cosine similarity. The directed graphs are converted into undirected graphs and the propagation is conducted on the converted undirected graphs by using a standard label propagation scheme. Spectral Graph Transducer is used in the proposed approach because of its effectiveness. The empirical study with several widely used datasets has shown the advantage of using directed graphs over some state-of-art semi-supervised techniques.

6.2 Future Work

The detailed description of the existing and proposed label propagation approaches is given with the empirical studies which have shown that label propagation can be effectively used in many applications. However, some problems need to be further studied.

Generally speaking, all label propagation approaches involve the graph which is associated with the similarity matrix and the propagation method which is usually determined by an optimization problem. How to define the appropriate similarity measure is always an important issue for label propagation. A number of research works have been focused on learning the best similarity matrix for different applications.

Besides computing the similarity matrix, there are also many problems involved in the propagation schemes of different models. In relation propagation framework, consider the multi-label classification case. Each node in the graph constructed from the data is a document-category pair. The problem arises when the number of documents and categories are large, which is usually the case in real world scenario. Although the approximation is proposed to alleviate the problem, how to scale the algorithm remains to be an issue. For rank propagation, the optimization problem is designed based on the loss function.

The definition of the loss function can have a significant influence on the performance as showed in the empirical study. In propagation over directed graphs, how to construct the directed graph can affect the performance heavily. Two asymmetric measures to construct directed graphs are proposed based on our assumptions. However, as the empirical study showed, if the assumptions are not consistent with the nature of the application, the directed graphs may not work well as expected. This problem can also be viewed as a similarity matrix learning problem.

BIBLIOGRAPHY

Bibliography

[1] B. T. Bartell, Cottrell G.W., and R.K. Belew. Learning the optimal parameters in a ranked retrieval system using multi-query relevance feedback. In *Proc. Symp. on document Analysis and Information Retrieval*, Las Vegas, April 1994.

[2] Brian Bartell, Garrison W. Cottrell, and Rik Belew. Learning to retrieve information. In *Current trends in connectionism: Proceedings of the Swedish Conference on Connectionism*, LEA: Hillsdale, 1995.

[3] Mikhail Belkin and Partha Niyogi. Using manifold structure for partially labeled classification. *Advances in Neural Information Processing Systems*, 2002.

[4] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International conf. on Machine Learning*, 2001.

[5] Avrim Blum, John Lafferty, Mugizi Robert Rwebangira, and Rajashekar Reddy. Semi-supervised learning using randomized mincuts. In *Proc. of ICML'04*, page 13, New York, NY, USA, 2004. ACM Press.

[6] C. Boutilier, R. Zemel, and B. Marlin. Active collaborative filtering. In *Proc. UAI*, 2003.

[7] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[9] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. Technical report, University College London, 2004.

[10] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *Proc. ICML*, 2005.

[11] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[12] K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of the conference on Neural Information Processing Systems(NIPS)*, 2001.

[13] Arthur Gretton Olivier Bousquet Dengyong Zhou, Jason Weston and Bernard Schölkopf. Ranking on data manifolds. *Advances in Neural Information Processing Systems*, (16), 2004.

[14] P. Doyle and J. Snell. *Random walks and electric networks.* Mathematical Assoc. of America, 1984.

[15] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In *Proceedings of the European Conference on Machine Learning*, pages 145 – 165, 2001.

[16] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proc. of the Fifteenth Intl. Conf. of Machine Learning*, pages 170–178, 1998.

[17] Rayid Ghani, Seán Slattery, and Yiming Yang. Hypertext categorization using hyperlink patterns and meta data. In Carla Brodley and Andrea Danyluk, editors, *Proceedings of ICML-01, 18th International Conference on Machine Learning*, pages 178–185, "Williams College, US", 2001. Morgan Kaufmann Publishers, San Francisco, US.

[18] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers, MIT Press*, 2000.

[19] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1997.

[20] T. Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning*, 1999.

[21] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, 1998.

[22] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[23] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

[24] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proc. ICML'03*, 2003.

[25] Klaus Brinker Kbrinker. Active learning of label ranking functions. In *Proc. 21st International Conf. on Machine Learning*, 2004.

[26] M. Kendall. Charles Griffin and Company Limited, 1948.

[27] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[28] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proc. 19th International Conf. on Machine Learning*, 2002.

[29] Oren Kurland and Lillian Lee. Pagerank without hyperlinks: Structural re-ranking using links induced by language models. In *Proc. of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 306–313, 2005.

[30] Neil D. Lawrence and Michael I. Jordan. Semi-supervised learning via gaussian processes. *Advances in Neural Information Processing Systems*, (17):753–760, 2005.

[31] G. Lebanon and J. Lafferty. Conditional models on the ranking poset. *Advances in Neural Information Processing Systems*, 15, 2003.

[32] Guy Lebanon and John Lafferty. Cranking:combining rankings using conditional probability models on permutations. In *International Conference on Machine Learning*, 2002.

[33] Tao Li, Chengliang Zhang, and Shenghuo Zhu. Empirical studies on mult-label classification.

[34] M. OConnor and J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of SIGIR-2001 Workshop on Recommender Systems*, 2001.

[35] Foster J. Provost and Tom Fawcett. Robust classification for imprecise environments. In *AAAI/IAAI,*, pages 706–713, 1998.

[36] J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proc. ICML*, 2005.

[37] S. E. Robertson and S. Walker. Okapi/keenbow at trec-8. In *Proc. TREC-8*, 1999.

[38] B. Pfahringer S. Kramer, G. Widmer and M. DeGroeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, 47:1 – 13, 2001.

[39] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[40] B. Sarwar, G. Karypis, and J. Konstan. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *5th International Conference on Computer and Information Technology*, 2002.

[41] M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[42] A. Shashua and A. Levin. Taxonomy of large margin principle algorithms for ordinal regression problems. Technical Report 39, Leibniz Center for Research, School of Computer Science and Eng., the Hebrew University of Jerusalem, 2002.

[43] Amnon Shashua and Anat Levin. Ranking with large margin principle: Two approaches. In *Proceedings of the 14th conference on Neural Infomation Processing System(NIPS)*, 2003.

[44] N. Srebro, J. D. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Proc. NIPS*, 2005.

[45] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. *Advances in Neural Information Processing Systems*, 14, 2001.

[46] Y Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, (13):2173 – 2200, 2001.

[47] C. Williams. Computation with infinite neural networks. *Neural Computation*, 10(5), 1998.

[48] G.-R. Xue, H.-J. Zeng, Z. Chen, W.-Y. Ma, H.-J. Zhang, and C.-J. Lu. Implicit link analysis for small web search. In *Proc. SIGIR '03*, pages 56–63, 2003.

[49] K. Yu, A. Schwaighofer, V. Tresp, W. Ma, and H. J. Zhang. Collaborative ensemble learning: combining collaborative and content-based information filtering via hierarchical bayes. In *Proc. UAI*, 2003.

[50] D. Zhou, B. Scholkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Proc. NIPS*, 2005.

[51] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16, 2004. MIT PRess, Cambridge.

[52] Dengyong Zhou, Jiayuan Huang, and Bernhard Scholkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML'05: Proceedings of the 22nd international conference on Machine Learning*, pages 1036–1043, New York, NY, USA, 2005. ACM press.

[53] S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labeled classification using maximum entropy method. In *Proc. ACM SIGIR*, 2005.

[54] Xiaojin Zhu. Semi-supervised learning learning literature survey. technical report 1530, University of Wisconsin-Madison, 2005.

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

[56] Xiaojin Zhu and John Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1052–1059, New York, NY, USA, 2005. ACM Press.

[57] Z.; Zhu, X.; Ghahramani and J. D. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proc. ICML*, 2003.