EXPLORING THE POWER OF HETEROGENEOUS INFORMATION SOURCES

BY

JING GAO

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair & Director of Research
Associate Professor ChengXiang Zhai
Professor Tarek Abdelzaher
Dr. Wei Fan, IBM T.J. Watson Research Center

Abstract

The big data challenge is one unique opportunity for both data mining and database research and engineering. A vast ocean of data are collected from trillions of connected devices in real time on a daily basis, and useful knowledge is usually buried in data of multiple genres, from different sources, in different formats, and with different types of representation. Many interesting patterns cannot be extracted from a single data collection, but have to be discovered from the integrative analysis of all heterogeneous data sources available. Although many algorithms have been developed to analyze multiple information sources, real applications continuously pose new challenges: Data can be gigantic, noisy, unreliable, dynamically evolving, highly imbalanced, and heterogeneous. Meanwhile, users provide limited feedback, have growing privacy concerns, and ask for actionable knowledge. In this thesis, we propose to explore the power of multiple heterogeneous information sources in such challenging learning scenarios. There are two interesting perspectives in learning from the correlations among multiple information sources: Explore their *similarities* (**consensus combination**), or their *differences* (**inconsistency detection**).

In consensus combination, we focuse on the task of classification with multiple information sources. Multiple information sources for the same set of objects can provide complimentary predictive powers, and by combining their expertise, the prediction accuracy is significantly improved. However, the major challenge is that it is hard to obtain sufficient and reliable labeled data for effective training because they require the efforts of experienced human annotators. In some data sources, we may only have a large amount of unlabeled data. Although such unlabeled information do not directly generate label predictions, they provide useful constraints on the classification task. Therefore, we first propose a graph based consensus maximization framework to combine multiple supervised and unsupervised models obtained from all the available information sources. We further demonstrate the benefits of combining multiple models on two specific learning scenarios. In transfer learning, we propose an effective model combination framework to transfer knowledge from multiple sources to a target domain with no labeled data. We also demonstrate the robustness of model combination on dynamically evolving data.

On the other hand, when unexpected disagreement is encountered across diverse information sources, this might raise a red flag and require in-depth investigation. Another line of my thesis research is to explore differences among multiple information sources to find anomalies. We first propose a spectral method to detect objects performing inconsistently across multiple heterogeneous information sources as a new type of anomalies. Traditional anomaly detection methods discover anomalies based on the degree of deviation from normal objects in one data source, whereas the proposed approach detects anomalies according to the degree of inconsistencies across multiple sources. The principle of inconsistency detection can benefit many applications, and in particular, we show how this principle can help identify anomalies in information networks and distributed systems. We propose probabilistic models to detect anomalies in a social community by comparing link and node information, and to detect system problems from connected machines in a distributed systems by modeling correlations among multiple machines.

In this thesis, we go beyond the scope of traditional ensemble learning to address challenges faced by many applications with multiple data sources. With the proposed consensus combination framework, labeled data are no longer a requirement for successful multi-source classification, instead, the use of existing labeling experts is maximized by integrating knowledge from relevant domains and unlabeled information sources. The proposed concept of inconsistency detection across multiple data sources opens up a new direction of anomaly detection. The detected anomalies, which cannot be found by traditional anomaly detection techniques, provide new insights into the application area. The algorithms we developed have been proved useful in many areas, including social network analysis, cyber-security, and business intelligence, and have the potential of being applied to many other areas, such as healthcare, bioinformatics, and energy efficiency. As both the amount of data and the number of sources in our world have been exploding, there are still great opportunities as well as numerous research challenges for inference of actionable knowledge from multiple heterogeneous sources of massive data collections.

To Lu and my parents.

Acknowledgments

I would like to express my great appreciation to all the people who gave me tremendous support and help during my Ph.D. study.

First of all, I would like to express my deep gratitude to my advisor, Prof. Jiawei Han, for his continuous support, guidance and encouragement. I benefited not only from his broad vision and direction about data mining research, but also his way of conducting rigorous research. I learnt from him how to identify key problems with impact, develop scientific approaches, and evaluate and present the ideas. His passion in research, teaching and student supervision has motivated me to pursue a career in academia, and I hope I will be a great advisor like him.

I also sincerely thank my other committee members: Dr. Wei Fan, Prof. ChengXiang Zhai and Prof. Tarek Abdelzaher. All of them not only provide constructive suggestions and comments on this thesis, but also offer numerous support and help in my career choice, and I am truly grateful for them. Dr. Wei Fan has been a great mentor to me over the past five years. His experience and vision in real data mining tasks has inspired me to take the challenge in multi-source data mining, and I learnt a great deal from the collaboration with him on many exciting topics. I learnt current information retrieval and text mining technology from Prof. ChengXiang Zhai's courses, and I was provided lots of useful feedback and suggestions from him during my PhD study. Prof. Tarek Abdelzaher has provided many exciting discussions, and helped me discover novel applications for the proposed approaches.

Many thanks to Prof. Feng Liang at Department of Statistics, for valuable discussions in constructing the right model for data mining problems as well as insightful suggestions on many of my research work. I would also like to thank my research collaborators, Mohammad Mehedy Masud and Prof. Latifur Khan, at University of Texas, Dallas. I benefited a lot from discussions and collaborations with them on data stream mining. Thanks to the support and help of Prof. Phillip

S. Yu at University of Illinois, Chicago on my study and career.

I have obtained valuable experiences in working on real-world data mining problems and products during summer internships. I would like to thank my mentors, collaborators and other members at Exploratory Stream Analytics Group at IBM T.J. Watson Research Center, and Robust & Secure Systems Group at NEC Labs America. Especially, I want to thank Dr. Olivier Verscheure, Dr. Deepak Turaga, Dr. Srinivasan Parthasarathy, Dr. Michael Wurst, Dr. Xiaoqiao Meng, Dr. Guofei Jiang, Dr. Haifeng Chen, Dr. Prem Melville and Dr. Rick Lawrence, who provided me the exciting opportunities of applying data mining techniques to real applications, and offered me critical suggestions and insightful perspectives on my research projects.

I would like to thank all professors, colleagues, and friends at Data and Information Systems (DAIS) lab, for their valuable discussions and help. In particular, I would like to thank the past and present members of data mining research group, including Deng Cai, Chen Chen, Hong Cheng, Marina Danilevsky, Hongbo Deng, Bolin Ding, Hector Gonzalez, Quanquan Gu, Manish Gupta, Ming Ji, Xin Jin, Hyung Sul Kim, Min-Soo Kim, Sangkyum Kim, Xiaolei Li, Zhenhui Li, Cindy Xide Lin, Chao Liu, Chandrasekar Ramachandran, Yizhou Sun, Lu An Tang, Chi Wang, Tim Weninger, Dong Xin, Xifeng Yan, Xiaoxin Yin, Zhijun Yin, Xiao Yu, Yintao Yu, Bo Zhao, Peixiang Zhao, and Feida Zhu. Also, many thanks to Prof. Kevin Chang, Prof. Julia Hockenmaier, and Liangliang Cao, Tao Cheng, Jing Jiang, Maryam Karimzadehgan, Yanen Li, Han Liu, Yue Lu, Qiaozhu Mei, Lin Tan, Ya Tang, Arash Termehchy, Nikita Spirin, Hongning Wang, Xuanhui Wang, Yang Wang, Duo Zhang for valuable discussions, collaborations and feedback on practice talks.

Many thanks to IBM for providing IBM PhD fellowship to support my PhD study. I also want to thank other institutions and funding agencies, such as National Science Foundation, Army Research Lab, Air Force Office of Scientific Research, and National Center for Supercomputing Applications, for their financial support and assistance. Thanks to Donna Coleman, Mary Beth Kelly, Colin Robertson and Jennifer Dittmar for providing administrative support.

Finally, I would like to thank my husband and my parents for their love, support and understanding. Without their encouragement and help, this thesis would be impossible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern technology has brought enormous benefits to almost all people, but at the same time makes our world a highly complex global system. Since the start of this decade, many issues have surfaced: the problems of global climate change and energy shortage, new security concerns ranging from identity theft to terrorism as well as imminent needs for better and more efficient healthcare systems. On the other hand, trillions of connected digital devices are producing a vast ocean of data. The flood of information comes from multiple heterogeneous information sources, and we need to discover knowledge from such huge amount of data quickly and effectively.

In this thesis, we explore the power of heterogeneous information sources in effective knowledge discovery. In this chapter, we first present motivating examples which cover a wide variety of applications, and discuss the major challenges in learning from multiple sources (Section 1.1). In Section 1.2, we summarize the problems, algorithmic contributions and applications of the proposed approaches.

1.1 Motivation

From the old story of "blind men and an elephant", we can learn the importance of capturing a big picture instead of focusing on one single perspective. In the story, a group of blind men touch different parts of an elephant to learn what it is like. Each of them makes a judgment about the elephant based on his own observation. For example, the man who touched the ear said it was like a fan while the man who grabbed the tail said it was like a rope. Clearly, each of them just got a partial view and did not arrive at an accurate description of the elephant. However, as they capture different aspects about the elephant, we can learn the big picture by integrating the knowledge about all the parts together.

In this thesis, we consider multiple data sources as different perspectives of observing the same set of objects. Depending on the specific data mining tasks, multiple information sources can refer to data collected from different places, data of different formats, or data that capture different aspects of the objects. For example, we can collect information about movies from many places, including movie review and rental websites (e.g., IMDB, Netflix, RottenTomato, Yahoo! Movies), or encyclopedia webistes (e.g., Wikipedia), or online social networks or forums (e.g., twitter), or fansites of movie stars. Such information describes different aspects of movies, including descriptions, cast, plots, reviews, tags, and involves different types of data, such as video, audio and text. To provide better web services or help film distributors in decision making, we need to conduct integrative analysis of all the information sources. For example, users' favorite movie genres can be inferred from their viewing history, ratings, reviews, or discussions on forums, and a knowledge integration framework that takes all these aspects into consideration usually achieves better results.

In numerous real-world applications, we have to combine and synthesize a variety of different information sources to make better decisions. Moreover, many interesting patterns cannot be extracted from a single perspective, but have to be discovered from the integrative analysis of all heterogeneous information sources available. We list a few more motivating applications as follows.

Social Media Analysis

With the explosive use of social networking, online video, digital photography and mobile phones, we have huge amount of information transferred across the globe everyday. As billions of individuals are using social media, numerous flows of user-generated data are captured today as never before. Such huge amount of data obtained from users reveals what they see, what they think and what they want. Numerous companies and organizations can benefit from the joint analysis of multiple channels of social media. For example, we want to find out user intent or behavior from their social networks, online blogs, reviews, and query logs for more effective customer targeting.

Cyber Security

The popularity of web services has attracted numerous spammers who try to comprise host machines with malicious software, such as viruses, worms, trojan horses and spyware. Moreover, new types of malicious activities are emerging with new applications on the Internet including online social networks, cloud computing and file sharing services. Therefore, machines connected to Internet are facing a significant risk. To detect security violations and spammers in a timely manner, we need to analyze network traffic, CPU usage, power consumption, software logs and many other information sources together.

Healthcare Systems

Many problems in the current healthcare systems: Rising costs, limited access, high error rates, lack of coverage, long response time, and lengthy development of new medicine, can be improved if we integrate the data and information from diagnosis, drug discovery, treatment, patient symptoms, medical bills, and health insurance plans. For example, to judge a person's health status more accurately, we have to make decision based on all of his health information including demographic characteristics, everyday lifestyle patterns, physiological data, and medical history.

Energy Efficiency

There has been an enormous increase in the demand for energy as a result of industrial development and population growth. It calls for less waste and more efficient use of energy. As sensors are put into home meters, pipelines and networks, we can collect a vast amount of heterogeneous energy management and usage data. Multi-source analysis techniques can turn the data into knowledge, which can help individuals, businesses, and utility companies make smarter decisions on how to consume energy and manage loads wisely.

As rich heterogeneous data can be collected in nearly every industry, people began to recognize the importance of integrating multiple data sources. In the field of data integration and data fusion, many algorithms have been developed to effectively integrate or combine raw data. Many studies focused on how to match the schemas of different data sources, detect entities that refer to the same real-world objects, answer queries by searching from multiple data sources, or combine multiple redundant information sources into one reliable source. In numerous applications that own multiple data sources, it is crucial not only to integrate or combine multiple data sources, but also consolidate different concepts for intelligent decision making. Therefore, in the field of knowledge integration, many algorithms have been developed to merge and synthesize models, rules, patterns obtained from multiple sources by reconciling their differences. These methods conduct classification or clustering from multiple data sources to identify more reliable and meaningful label predictions or clusters from the joint analysis of multiple information sources. An overview of existing data integration and knowledge integration methods can be found in Chapter 2.

Despite these efforts, numerous critical challenges still remain unaddressed. Data are becoming heterogeneous, large-scale, noisy, incomplete, highly imbalanced and dynamically evolving. Moreover, lack of supervision, growing privacy concerns and demands for actionable and interpretable knowledge add to the complexity. In particular, we focus on the following challenges in this thesis:

- *Incompatible formats.* Unleashing the full power of multiple information sources is a very challenging problem when schemas used to represent each data collection are different (heterogeneous information sources). For example, information regarding a patient's symptoms can be found from his lab test results, physician notes and ultrasound images.

- *Lack of groundtruths.* The main bottleneck of many learning algorithms is that a large, often prohibitive, number of labeled data are needed to build an accurate classifier. In many disciplines, this cannot be achieved due to the high cost of manual labeling.

- *Concept Evolution.* Many applications generate continuously arriving data, whose distribution is evolving in an unforseen way. For example, both normal network traffic features and spam characteristics evolve over time. It is quite difficult to make accurate predictions on such evolving data.

- *Imbalanced Distribution.* Skewed class distributions in classification can cause serious problems for many learning algorithms where the minority class is usually ignored. However, misclassifying minority objects invokes a higher cost, for example, misclassify an infected machine to be a normal host can cause serious damage to the computer network.

- *Incomplete and Noisy Data.* Due to the nature of noisiness, inconsistency, dynamics, and inter-dependency of the physical world data, each data source could contain glitches, inconsistencies, errors, and missing values.

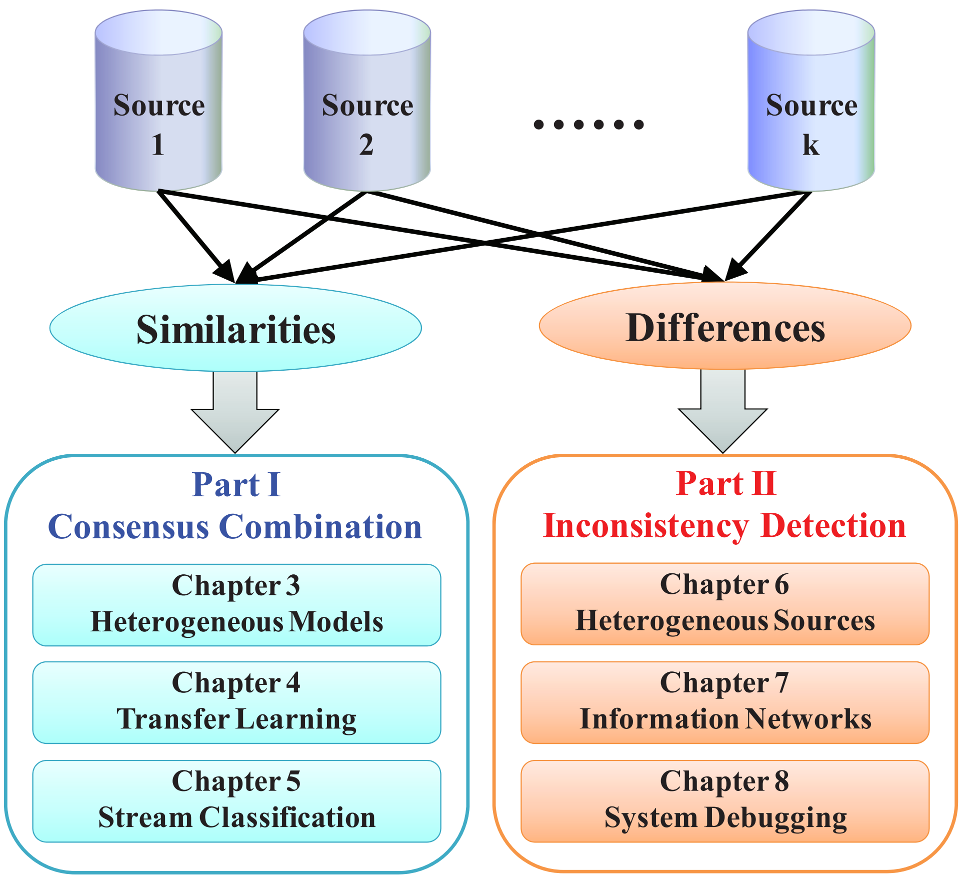


Figure 1.1: A Roadmap of Our Work in Mining Multiple Information Sources

1.2 Thesis Summary

In this thesis, we propose to explore the power of multiple heterogeneous information sources in challenging learning scenarios. Figure 1.1 shows a roadmap of our work. There are two interesting perspectives in learning from the correlations among multiple information sources: Explore their *similarities* (**consensus combination**), or their *differences* (**inconsistency detection**).

Part I: Consensus Combination

In this part, we focus on the task of classification with multiple information sources. Multiple information sources for the same set of objects can provide complimentary predictive powers, and by combining their expertise, the prediction accuracy is significantly improved. We propose a graph based consensus maximization framework to combine multiple supervised and unsupervised models, and also develop model combination algorithms for two specific learning scenarios.

Consensus Combination of Multiple Heterogeneous Models (Chapter 3). Ensemble learning has emerged as a powerful method for combining multiple models. However, due to the high costs of manual labeling, it is hard to obtain sufficient and reliable labeled data for effective training. Meanwhile, lots of unlabeled data exist in these sources, and we can readily obtain multiple unsupervised models. Although unsupervised models do not directly generate a class label prediction for each object, they provide useful constraints on the joint predictions for a set of related objects. Therefore, incorporating these unsupervised models into the ensemble of supervised models can lead to better prediction performance. We study ensemble learning with outputs from multiple supervised and unsupervised models, a topic where little work has been done. We propose to consolidate a classification solution by maximizing the consensus among both supervised predictions and unsupervised constraints. We cast this ensemble task as an optimization problem on a bipartite graph, where the objective function favors the smoothness of the predictions over the graph, but penalizes the deviations from the initial labeling provided by the supervised models. We solve this problem through iterative propagation of probability estimates among neighboring nodes and prove the optimality of the solution. With the proposed framework, labeled data are no longer a requirement for successful multi-source classification, instead, the use of existing labeling experts is maximized by integrating knowledge from relevant domains and unlabeled information sources.

Consensus Combination for Transfer Learning (Chapter 4). In many applications, it is expensive or impossible to collect enough labeled data for accurate classification in the domain of interest (target domain), however, there are abundant labeled data in some relevant domains (source domains). For example, when training a spam filter for a particular user, if we don't have any labeled data from the user for training, we can only rely on spam and ham emails from multiple public resources. Therefore, we want to transfer knowledge from multiple relevant source domains to the target domain. We notice that different classifiers trained from these source domains contain different knowledge about the target domain and thus have different advantages. The challenge is how to dynamically select the model that best represents the true target distribution underlying each example in the target domain. To solve this problem, we propose a locally weighted ensemble framework to adapt useful knowledge from multiple source domains to the target domain. The weights of source domains are dynamically assigned according to each domain's predictive power on each target object by comparing neighborhood graphs constructed from source and target domains. We show that the proposed method can successfully identify the knowledge from source domains that is useful to predict in the target domain and transfer such information to the target domain.

Consensus Combination for Stream Classification (Chapter 5). Many real applications generate continuously arriving data, known as data streams. To help decision making, we want to correctly classify an incoming data record based on the model learnt from historical labeled data. The challenge is the existence of distribution evolution or concept drifts, where one actually may never know either how or when the distribution changes. We propose a robust model averaging framework combining multiple supervised models, and demonstrated both formally and empirically that it can reduce generalization errors and outperform single models on stream data. We also consider a more challenging situation in stream data classification, where the class distributions in the data are skewed, i.e, there are few positives but lots of negatives, such as network intrusions (positives) and normal records (negatives) in network traffic. By estimating posterior probabilities using an ensemble of models to match the distributions over under-samples of negatives and repeated samples of positives, the proposed framework can significantly improve the reliability of label predictions on the more important positive class (e.g., the intrusions).

Part II: Inconsistency Detection

Integrating multiple models gives us the gains in classification performance. On the other hand, by exploring the differences among sources, we can identify something unusual and interesting, and thus another line of this thesis is to detect anomalies or inconsistencies across multiple information sources. We propose a general framework to detect inconsistencies across multiple heterogeneous information sources, as well as approaches to find objects performing inconsistently in information networks and distributed systems.

Inconsistency Detection across Multiple Heterogeneous Sources (Chapter 6). We propose to detect objects that have inconsistent behavior among multiple heterogeneous sources. On each of the information sources that describe a set of objects, a relationship graph can be derived to characterize the pairwise similarities between objects where the edge weight indicates the degree of similarity. Clearly, objects form a variety of clusters or communities based on each similarity relationship. However, there are some objects that fall into different clusters with respect to different sources. Such objects which perform inconsistently can be regarded as anomalies. For example, there exist movies that are expected to be liked by kids by genre, but are liked by grown-ups based on user viewing history. To identify such objects, we compute the distance between different eigen decomposition results of the same object with respect to different sources as its anomalous score, and give interpretations from the perspectives of constrained spectral clustering and random walks over graph. Inconsistency detection can detect anomalies that cannot be found by traditional anomaly detection techniques and provide new insights into the application area. The proposed approach can benefit many different fields where such diverse information sources are available to capture object properties and similarity relationships.

Inconsistency Detection for Information Networks (Chapter 7). Linked or networked data are ubiquitous in many applications. Examples include web data or hypertext documents connected via hyperlinks, social networks or user profiles connected via friend links, co-authorship and citation information, blog data, movie reviews and so on. In these datasets (called information networks), closely related objects that share the same properties or interests form a community. For example, a community in blogsphere could be users mostly interested in cell phone reviews and news. Outlier detection in information networks can reveal important anomalous and interesting behavior that are not obvious if community information is ignored. An example could be a low-income person being friends with many rich people even though his income is not anomalously low when considered over the entire population. We introduce the concept of community outliers and propose an efficient solution by modeling networked data as a mixture model composed of multiple normal communities and a set of randomly generated outliers. The probabilistic model characterizes both data and links simultaneously by defining their joint distribution based on hidden Markov random fields (HMRF). Maximizing the data likelihood and the posterior of the model gives the solution to the outlier inference problem. Experimental results demonstrate importance of this concept as well as the effectiveness and efficiency of the proposed approach.

Inconsistency Detection for System Debugging (Chapter 8). In today's large-scale distributed systems, the same type of information may be collected from each machine in the system. Although some knowledge can be extracted from each individual information source, a much richer body of knowledge can only be obtained by exploring the correlations or interactions across different sources. For example, the correlations between measurements collected across the distributed system can be used to infer the system behavior, and thus a reasonable model to describe such correlations is crucially important in detecting and locating system problems. We propose a transition probability model to characterize pairwise measurement correlations. Different from existing methods, the proposed solution can discover both the spatial (across system measurements) and temporal (across observation time) correlations, and thus such a model can successfully represent the system normal profiles. Whenever a record cannot be explained by the correlation model, it represents an anomaly. The effectiveness of this framework is demonstrated in its ability of detecting anomalous events and locating problematic sources from real monitoring data of three companies' infrastructures.

Finally, we conclude this thesis by summarizing our contributions and discussing future directions in Chapter 9. In summary, we have proposed important learning problems motivated by real challenges, and contributed several key principles and algorithms to the field of multi-source learning:

- We demonstrate the benefits of analyzing multiple information sources simultaneously from both theoretical and experimental perspectives. The advantages of multi-source mining are shown in a variety of difficult learning scenarios.

- To solve each problem, we transform heterogeneous information sources to a robust representation summarizing the key information, formulate the problem as an optimization problem or a probabilistic model, and solve it using the combination of multiple techniques, such as block coordinate descent, eigen decomposition, iterated conditional modes, and expectation maximization techniques. We also give interpretations of the proposed methods from other perspectives including random walks, spectral embedding and spectral clustering.

- We have developed several core algorithms for integrating knowledge from multiple data sources. We systematically study the consensus combination problem by proposing a graph based approach to combine multiple supervised and unsupervised models, a locally weighted ensemble framework for transfer learning, and a model averaging approach for stream data classification. We propose to detect inconsistency across multiple sources by developing a spectral framework that capture source-wise inconsistencies, a probabilistic model of community outliers in information networks and a statistical model of measurement correlations in distributed systems.

As demonstrated in many problems, the proposed algorithms can extract key knowledge from multiple heterogeneous sources. For example, our algorithms have been effectively applied to the following applications:

- *Social Media.* The consensus combination methods combine heterogeneous channels of information and thus provide robust and accurate solutions to social media analysis. Our proposed approaches achieve high accuracy, efficiency, and scalability on a variety of applications including document categorization, sentiment analysis, researcher profiling, and movie recommendation. Also, we have identified meaningful anomalies from publication networks and movie networks through inconsistency detection.

- *Cyber-security.* Consensus combination techniques have been used to integrate heterogeneous anomaly detectors into a more robust detector. The stream ensemble model is demonstrated useful in detecting network intrusions and malware. Inconsistency detection across multiple machines is successfully applied on distributed systems to detect system-wide problems.

Besides these applications, the algorithms have the potential of being applied to many different fields including healthcare, bioinformatics, business intelligence and energy efficiency. This thesis show that learning from multiple information sources **simultaneously** is the key solution to the effective knowledge acquisition.

Chapter 2

An Overview of Mining Multiple Information Sources

Recent technology developments have made it possible and affordable for us to gather and store a huge amount of data from multiple sources. In the past, researchers proposed many approaches to handle multiple information sources.

We summarize various learning problems in Table 2.1. One dimension represents the data mining tasks studied in this thesis, including supervised learning (classification), unsupervised learning (clustering), semi-supervised learning, and anomaly detection. The methods developed for each data mining task are either based on single-model (or single-source), or multi-model (multi-source). As shown in this table, the two general learning frameworks proposed in this thesis, e.g., consensus maximization and inconsistency detection, can be regarded as a multi-source semi-supervised learning approach and a multi-source anomaly detection approach respectively. Little work has been done on these two topics.

In this chapter, we give an overview of the methods presented in Table 2.1. We first briefly introduce single-model data mining techniques in Section 2.1. Then we review existing work on raw data level integration: Data integration and data fusion, in Section 2.2. After that, we discuss multi-source classification and clustering in Sections 2.3 and 2.4 respectively. In Section 2.5, we discuss some other studies in the general field of multi-source mining.

2.1 Single-model Data Mining Methods

Many efforts have been devoted to developing single-model data mining algorithms. In this section, we review these techniques for classification, clustering, semi-supervised learning and anomaly detection.

Table 2.1: An Overview of Single-Model and Multi-Model Mining Algorithms

	Classification	Clustering	Semi-Supervised Learning	Anomaly Detection
single-model	SVM Logistic Regression ...	K-means Spectral Clustering ...	Graph-based Transductive SVM ...	Distance-based Density-based ...
multi-model	Ensemble Learning Multi-view Learning ...	Clustering Ensemble Multi-view Clustering ...	**Consensus Combination**	**Inconsistency Detection**

Classification Classification, or supervised learning, tries to infer a function that maps feature values into class labels from training data, and apply the function to data with unknown class labels. The function is called a model or classifier. In general, a supervised learning algorithm defines a hypothesis space to search for the correct model as well as the generalization error of the model on future test data, and search for the model that minimizes generalization error in the space. A variety of classification algorithms have been developed [22], including Support Vector Machines, linear regression, logistic regression, Naive Bayes, decision trees, k-nearest neighbor algorithm, and Neural Networks, and they differ in the hypothesis space and generalization error formulation.

Clustering Clustering, or unsupervised learning, is the task of partitioning a set of objects into multiple clusters so that objects within a cluster are similar to each other while objects in different clusters are dissimilar [92]. As the notion of similarity varies among different clustering algorithms, the clusters found by different algorithms can be quite different in their properties. Some well-known clustering algorithms include hierarchical clustering, partitional clustering (e.g., K-means), density-based clustering (e.g., DBSCAN, OPTICS), and spectral clustering algorithms [82].

Semi-supervised learning Recent studies reveal that unlabeled information can be combined with labeled information to improve the accuracy of supervised learning, which leads to semi-supervised and transductive learning. The study of semi-supervised learning [187] is motivated by the fact that labeled data are hard to acquire but there are usually plenty of unlabeled data. Some widely used semi-supervised learning methods include EM with generative mixture models [39], transductive support vector machines [96, 189], and graph-based methods [186, 182]. Let x and y denote the feature vector and the class label respectively. In these methods, the unlabeled information is usually taken into account in estimating $P(x)$, which is used to influence the estimation of

$P(y|x)$. Some assumptions need to be made, for example, many semi-supervised learning methods assume that the classification boundary reside in areas with low $P(x)$.

Different from the proposed consensus maximization framework, semi-supervised learning (SSL) algorithms only take one supervised source (i.e., the labeled objects) and one unsupervised source (i.e., the similarity graph), and thus it cannot be applied to combining multiple models. Some SSL methods [78] can incorporate results from an external classifier into the graph, but obviously they cannot handle multiple classifiers and multiple unsupervised sources. To apply standard SSL algorithms on our problem, we must first fuse all supervised models into one by some ensemble approach, and fuse all unsupervised models into one by defining a similarity function. Such a compression may lead to information loss, whereas the proposed method retains all the information and thus consensus can be reached among the outputs of all the base models.

Anomaly detection Anomaly detection, sometimes referred to as outlier detection or novelty detection, is the procedure of identifying aberrant or interesting objects whose characteristics deviate significantly from the majority of the data. It is widely used in a variety of domains, such as intrusion detection, fraud detection, health monitoring, and so on. It is an important task because anomalies usually represent significant and critical points, such as intrusions found from network traffic, identity theft in credit card transactions and tumors detected from medical images. Therefore, the problem of anomaly detection has been widely studied and existing methods can be roughly divided into the following categories [31]: 1) Model-based techniques [48, 145] where a classifier or a clustering model is learnt to model normal behavior; 2) Proximity-based techniques [104, 27, 120] where objects that are far away from their neighbors are detected as anomalies; 3) Statistical methods [126] where a statistical model is used to fit majority of the data and anomalous objects are assumed to occur in low probability regions; 4) Information theoretic techniques [4] which identify a set of objects that induce irregularities in the information content of the data set; 5) Spectral techniques [89, 110] which project objects into a low-dimensional space so that normal and anomalous objects are easy to separate.

The techniques discussed above primarily focus on identifying objects that are dissimilar to most of the other objects from a *single* data source. In the proposed inconsistency detection framework, we aim at detecting objects that have “inconsistent behavior” across multiple information sources.

The detected anomalies can exhibit normal behavior judging from one data source, but perform abnormally when looking across multiple sources. This definition differs from that of traditional anomaly detection, and thus provides a novel perspective to the problem of detecting critical exceptions from data.

2.2 Data Integration and Data Fusion

The goal of data integration is to combine data from multiple sources and provide users a unified view. Most of the data integration research focuses on matching schemas of multiple sources, and conduct query processing from multiple sources [80, 45]. Data integration system [112] can be regarded as a triple $< G, S, M >$ where G is the global schema, S is the heterogeneous set of source schemas, and M is the mapping that maps queries between the source and the global schemas. Besides schema-level matching, people also investigated how to match the same entity across different data sources. Entity resolution [106, 171, 19, 108], sometimes referred to as object matching, duplicate identification, record linkage, or reference reconciliation, is the task of identifying the same real-world objects from multiple sources by evaluating similarities among objects or learning from labeled entity pairs that have been matched. Another important issue in multiple source data integration is the quality of data sources [10]. Methods have been developed to identify trustworthy information sources when integrating the facts obtained from the available information sources [177].

While data integration focuses on matching schemas to assist query processing over multiple databases, data or information fusion combines multiple redundant noisy information sources into a more reliable source [81, 141]. There are different levels of fusion. For example, data-level fusion combines several sources of raw data into a more robust representation, whereas decision-level fusion only integrates decisions obtained from multiple sources. Our proposed consensus maximization framework can be regarded as a decision-level fusion method, but different from existing work, we are the first to propose to combine both supervised and unsupervised information sources.

2.3 Ensemble and Multi-view Learning

In supervised learning, classification ensemble approaches [12, 44, 137, 109, 147] train multiple models from training data and combine their predictions. Although ensemble learning methods are not explicitly dealing with multiple data sources, they share a similar principle with multi-source learning: Combine complementary predictive power of multiple models is superior to using one single model. There are two critical components in ensemble learning: Training base models and learning their combinations. The goal is to derive diversified base models and minimize the generalized error by combining these models. Some of the methods, e.g., boosting [55], bayesian model averaging [86], and rule ensemble [56], learn both the base models and the combination from the labeled data, while the others including bagging [25], random forests [26], and random decision tree [51], train multiple base classifiers from the labeled data but combine the base models through majority voting. Methods such as mixture of experts [91] and stacked generalization [173] try to obtain a meta-learner on top of the model output by using the labels of the raw data as feedback. In [30], selection of base models in ensemble learning is studied. Ensemble learning is demonstrated useful in many data mining competitions (e.g., Netflix contest[1], KDD cup[2], ICDM contest[3]) and real-world applications. The methods reviewed above focus on combining models learnt from a single data source, instead of synthesizing knowledge from multiple sources.

The emergence of multi-source applications motivate the study of learning from data with multiple views where each view describes one aspect of the objects. In multi-view learning, a joint model is learnt from both labeled and unlabeled data of multiple sources. Most of the work focuses on the scenarios with two views. The basic assumption is that a small amount of labeled data and lots of unlabeled data are available, and there exist two independent views with compatible target functions for the task of classification. We can learn two classifiers from the two views and reduce the search space to where the two classifiers agree. The proposed methods either search for the compatible hypothesis explicitly [23, 132] or search in a reduced space where the correlations of two views are maximized [53, 185], or use a regularization term to enforce that multiple classifiers agree with each other on the unlabeled data [37, 151, 57, 124, 3]. These approaches exploit multiple

[1]http://www.netflixprize.com/
[2]http://www.kddcup-orange.com/
[3]http://www.cs.uu.nl/groups/ADA/icdm08cup/

redundant views to effectively learn from unlabeled data. Because the weaknesses of one view complement the strengths of the other, multi-view learning has been shown to be advantageous when compared to learning with only a single view. However, multi-view learning cannot be easily extended to the cases where multiple sources have multiple semantics and formats. In contrast, our proposed consensus maximization framework, which synthesizes high-level knowledge obtained from multiples sources, can be applied to numerous applications with heterogeneous information sources.

All these studies aim at solving *one* problem, where multiple information sources can contribute to the task. On the contrary, multi-task learning [29, 40] deals with multiple tasks simultaneously by exploiting dependence among tasks, which has a different problem setting from this thesis.

2.4 Clustering Ensemble and Multi-view Clustering

Similar to the supervised learning case, we first discuss related work on combining multiple unsupervised clustering models. In unsupervised learning, many clustering ensemble methods [74] have been developed to find a consensus clustering from multiple partitionings. The research focus is to derive multiple clustering partitions and combine their output to minimize the disagreement. Different from supervised ensembles, in unsupervised ensembles, the correspondence between clusters in different clustering solutions is unknown, and thus the number of possible clustering solutions is exponential. Existing clustering ensemble approaches differ in the choices of consensus definitions and the representation of the base model output. The base model outputs are usually represented as categorical features, but they can also be summarized in a graph [155, 54] or a three-dimensional array [152]. There exist some generative approaches which try to maximize the likelihood of base model output [138, 163, 164], whereas most of the methods formulate consensus clustering as an optimization problem using information-theoretic [155], median partition [122, 116] or correlation clustering [75] objective functions.

Clustering ensemble methods do not work on multi-source data directly. To conduct clustering on data with multiple views, people explore the joint analysis of multiple views to compute a global clustering solution [21, 183, 121, 33]. The basic principle is that the same object should be assigned to the same cluster under different views. These approaches either extract a set of shared features from the multiple views and then apply any traditional clustering algorithm, or exploit the multiple views of the data as part of the clustering algorithm by utilizing information from the other view to enhance clustering of each view. With the assumption that objects share the same clustering structure across views, multi-view clustering can identify such shared clustering more effectively than clustering two views separately. Note that both clustering ensemble and multi-view clustering aim at conducting clustering, whose goal is thus different from the proposed consensus maximization and inconsistency detection problems.

The ensemble techniques have been mostly studied in supervised and unsupervised learning communities separately, besides some general reviews [134]. In this thesis, we propose to combine the power of all sources when both supervised and unsupervised models are available for a single task and show that such integration leads to better performance. We also investigate the problem of comparing multiple sources to identify inconsistencies across sources.

2.5 Other Multi-Source Mining Studies

Multiple information sources contain rich knowledge. Besides classification and clustering from multiple sources, there exist studies on other aspects of multi-source mining. In the field of pattern mining, most efforts have been devoted to comparing frequent patterns across multiple sources. The patterns of interest are the ones that have the most significant difference among multiple data sets. Specifically, emerging patterns or contrast patterns [46, 13, 168] are defined as itemsets whose supports increase significantly from one source/class to another. Most of the work in this field deal with transaction datasets and their goal is to identify interesting rules through support comparisons. Wang *et al.* [161] consider the problem of ranking in heterogeneous domains where preference information from a source domain can be transferred to the target domain for more effective ranking. In [113], a cross-domain collaborative filtering technique was proposed to transfer knowledge from one domain to another domain to alleviate the rating sparsity problem. Metric learning was also studied in the scenario of multiple data sources where a joint embedding projection is learnt from multiple views of data [83, 139].

Part I

Consensus Combination

Chapter 3

Consensus Combination of Multiple Heterogeneous Models

Given models obtained from multiple heterogeneous sources, how can we effectively combine them for more accurate label predictions? In this part, we present our studies on combining multiple sources through exploring their *similarities*. There usually exist multiple information sources contributing to the classification task, and they provide complementary knowledge and we should combine their expertise for better predictions. In this chapter, we propose and solve a novel problem of consensus maximization among multiple supervised and unsupervised models. In Chapters 4 and 5, we present two targeted solutions of model combination for transfer learning and stream classification scenarios.

The success of ensemble techniques has been proven theoretically and observed in real practice. However, the major challenge is that it is hard to obtain sufficient and reliable labeled data for effective training because they require the efforts of experienced human annotators. To tackle this challenge, we study the problem of consolidating multiple supervised and unsupervised information sources by negotiating their predictions to form a final superior classification solution [67, 68]. The proposed method can utilize all the available sources in the consensus combination framework, no matter they contain labeled or unlabeled information. A global optimal label assignment for objects is derived by maximizing consensus among the given models. This consensus maximization approach crosses the boundary between supervised and unsupervised learning, and its effectiveness has been shown in many real-world problems, where the classification accuracy is significantly improved. In particular, the proposed method has been used to solve the following problems: 1) user-generated video classification based on video, audio, and text features [140]; 2) decision fusions of heterogeneous sensor nodes in sensor networks [156]; and 3) combination of heterogeneous anomaly detectors to improve performance over botnet or network traffic anomaly detection in cyber-security areas [65].

Table 3.1: Predicting Users' Favorite Movie Types

Person	Supervised				Unsupervised				Consensus
	M_1	M_2	...	M_r	M_{r+1}	M_{r+2}	...	M_m	
Mary	Comedy	Comedy	...	Thriller	Cluster2	Cluster1	...	Cluster3	Comedy
Jack	Comedy	Comedy	...	Comedy	Cluster2	Cluster2	...	Cluster3	Comedy
Lucy	Comedy	Thriller	...	Comedy	Cluster1	Cluster3	...	Cluster1	Comedy
Mike	Thriller	Thriller	...	Thriller	Cluster3	Cluster1	...	Cluster1	Thriller
Jim	Action	Thriller	...	Action	Cluster3	Cluster2	...	Cluster1	Thriller
Bob	Action	Action	...	Action	Cluster1	Cluster1	...	Cluster2	Action
Tom	Thriller	Comedy	...	Action	Cluster3	Cluster1	...	Cluster2	Thriller

3.1 Overview

Suppose there are a set of objects that need to be classified into corresponding categories. The predictive information comes from multiple data sources, each of which either transfers label information from relevant domains (supervised classification), or derives grouping constraints from the unlabeled target objects (unsupervised clustering). Since these models are derived from diversified and heterogeneous sources, the strength of one usually complements the weakness of the other, and thus maximizing the agreement among them can significantly boost the performance. We illustrate how multiple sources provide "complementary" expertise and why their consensus combination produces more accurate results through a real example.

Suppose we have seven users and they are using a movie recommendation service. Our goal is to predict each user's favorite movie type as one of {comedy, thriller, action}. Naturally, the first data source for the task involves the movies they have watched, which can be collected from movie review or rental websites. If Mary watched more comedy movies than the other types of movies, we can say that her favorite movie type is comedy. If Bob watched a lot of action movies, his favorite is action. In general, we can infer a user's favorite movie type as the type of the movies he's watched the most often. However, we only have limited access to the list of movies users have seen in practice. A user might have watched 100 movies, but we only know 20 of them. Then the decision made based on such partial information will contain errors. We summarize the decisions made by this movie type data source in Column M_1 of Table 3.1. We can make such predictions based on some other data sources about the users and the movies, such as user taggings and personal information of users. Each of them gives a useful but not perfect classification solution. We summarize the decisions made by such supervised data sources in Columns M_1 to M_r of Table

3.1. M_i denotes a supervised model obtained from the i-th data source, and the corresponding column shows the prediction results of applying M_i on the seven users.

We can also collect information about how users interact with each other regarding movies from some movie discussion forums or social networks. Based on this data source, we can partition users into several clusters based on their relationships, as shown in Column M_{r+1} of Table 3.1. For example, Mary and Jack interact a lot on movies. Thus they are put into the same cluster, and they are more likely to like the same type of movies. This data source provides useful constraints on users' favorite movie type predictions. We should try to assign the same movie type to users in one cluster. However, the clustering constraint may not always be correct because sometimes users interact on issues not related to movie types, for example, movie theaters or movie stars. Again, it is a useful but not perfect data source. We can receive multiple clustering solutions from multiple such sources, and we summarize the results obtained from these unsupervised data sources in Columns M_{r+1} to M_m of Table 3.1. Each of them clusters the seven users into three clusters and provides the constraints.

In summary, we have multiple data sources for the task of movie recommendation. Due to distributed computing, privacy preserving or knowledge reuse reasons, we don't have access to raw data of each source, instead, each source just provides labeling or grouping results on the seven users. A labeling solution simply predicts each user's favorite movie type, whereas a clustering solution gives the constraint. Each of the solutions is derived based on partial information. However, these solutions are based on heterogeneous sources and capture different aspects about the label predictions of the target objects. Therefore, they do complement each other. Given all these solutions, our goal is to combine them into a consolidated solution (as shown in the last column of Table 3.1), which leverages different information sources, gives the global picture and thus is more accurate than each base solution derived from a single source.

Similar situation exists in many applications. When we predict people's occupations, we can integrate information from multiple social network websites. To predict whether a user will buy a product, we can look at his purchase history, personal information and social networks. When we predict a researcher's area, the information we have include the journals/conferences he published in, the keywords of the publications and co-authorship information. In all these examples, since multiple information sources give complementary labeling and clustering solutions, we seek to integrate knowledge from these sources for better predictions.

Formally, we consider the general problem of combining the *outputs* of multiple supervised and unsupervised models to improve prediction accuracy. Suppose we have a set of objects $X = \{x_1, x_2, \ldots, x_n\}$ from c classes. There are m models that provide information about the classification of X. The first r of them are (supervised) classifiers, and the remaining are (unsupervised) clustering models. The objective is to predict the class label of $x_i \in X$, which agrees with the base classifiers' predictions, and meanwhile, satisfies the constraints enforced by the clustering models as much as possible. We refer to this problem as *consensus maximization*.

Ensemble methods have been studied in supervised learning [12, 137, 147] and unsupervised learning [74] communities separately, and thus existing ensemble methods cannot be used to combine multiple supervised and unsupervised models. Multi-view learning algorithms [37, 53, 151, 57] require the access to raw data, and thus cannot handle the cases where only high-level concepts or models are available. Semi-supervised learning [187] algorithms utilizes unlabeled information together with labeled information to improve classification accuracy. However, they cannot be applied to the cases with multiple information sources. In [63], we proposed a heuristic method to combine heterogeneous information sources. In this chapter, we introduce the concept of consensus maximization and solve the problem over a bipartite graph representation. The proposed consensus maximization problem is a challenging problem. First of all, it cannot be solved by simple majority voting because the correspondence between the cluster ID and the class label is unknown, and the same cluster ID in different clustering models may represent different clusters. Secondly, to achieve maximum agreement among various models, we must seek a global optimal prediction for the target objects. With n objects and c classes, there are c^n possible label assignments and thus the search space is exponential.

To tackle this problem, we propose to summarize the base model outputs using a bipartite graph in a lossless manner. To reach maximum consensus among all the models, we define an optimization problem over the bipartite graph whose objective function penalizes the deviations from the base classifiers' predictions and the discrepancies of the predicted class labels among nearby nodes. We solve the optimization problem using a coordinate descent method and derive

Table 3.2: Important Notations

Symbol	Definition
$1, \ldots, c$	class indexes
$x_1, \ldots, x_n$	objects
$g_1, \ldots, g_s$	groups from supervised models
$g_{s+1}, \ldots, g_v$	groups from unsupervised models
$A_{n \times v} = [a_{ij}]$	a_{ij}-indicator of object i in group j
$U_{n \times c} = [u_{iz}]$	$u_{iz} = P(z \mid x_i)$-probability of object i wrt class z
$Q_{v \times c} = [q_{jz}]$	$q_{jz} = P(z \mid g_j)$-probability of group j wrt class z
$Y_{v \times c} = [y_{jz}]$	y_{jz}-indicator of group j predicted as class z

a global optimal label assignment for the target objects, which is different from the result of traditional majority voting and model combination approaches. The proposed method can be applied to more complicated situations. For example, it can work in the scenario when each object has a probabilistic label assignment, or each object has multiple labels to describe its category, or some predictions are missing from each source. The method can further be adapted to handle imbalanced class distributions and label feedback integration.

The rest of the chapter is organized as follows. In Section 3.2, we formally define the graph-based consensus maximization problem and propose an iterative algorithm to solve it, which propagates label information among neighboring nodes until stabilization. We prove the optimality of the proposed solution in Section 3.3. We also present two different interpretations in Section 3.4. We discuss how to incorporate feedback obtained from a few labeled target objects into the framework and how to handle imbalanced class distributions in Section 3.5. An extensive experimental study is carried out in Section 3.6, where the benefits of the proposed approach are illustrated on 20 Newsgroup, Cora research papers, DBLP bibliography and multi-domain sentiment data sets.

3.2 Methodology

Suppose we have the outputs of r classification models and $(m-r)$ clustering models on a set of objects $X = \{x_1, \ldots, x_n\}$. Note that each x_i can have multiple sources to describe it, and thus each clustering model can be learnt from one of the sources without accessing the other data sources. For the sake of simplicity, we assume that each object is assigned to only one class or cluster by each of the m models, and the number of clusters in each clustering model is c, the same as the number

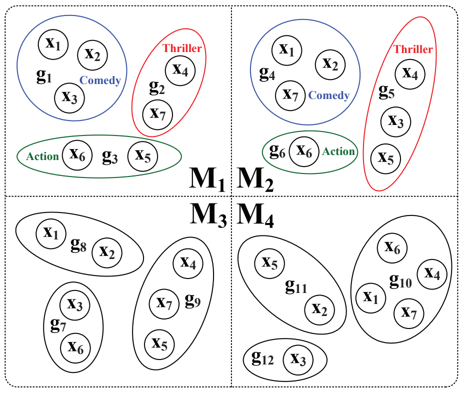


Figure 3.1: Groups-Objects Relationship

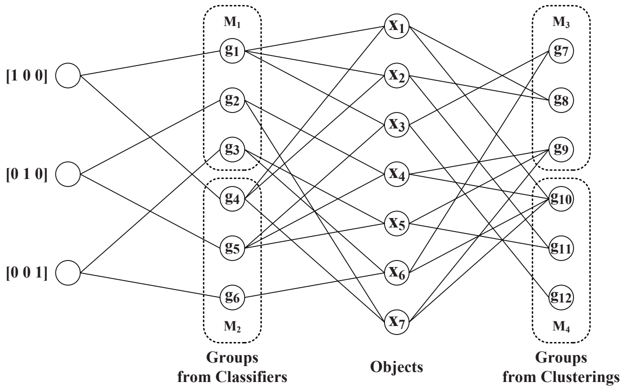


Figure 3.2: Bipartite Graph Representation

of classes. Note that cluster ID z may not be related to class z. Each base model partitions X into c groups, and there are a total of $v = mc$ groups, where the first $s = rc$ groups are generated by classifiers and the remaining $v - s$ groups are generated by clustering algorithms. Before proceeding further, we introduce some notations that will be used in the following discussion: $B_{n\times m}$ denotes an $n \times m$ matrix with b_{ij} representing the (ij)-th entry, and $\vec{b}_{i\cdot}$ and $\vec{b}_{\cdot j}$ denote vectors of row i and column j, respectively. See Table 3.2 for a summary of important symbols.

Each model, supervised or unsupervised, partitions X into groups, and objects in the same group share either the same predicted class label or the same cluster ID. We can represent the objects and groups in a bipartite graph, where we have two types of nodes: the *object nodes* $x_1, \ldots, x_n$ and the *group nodes* $g_1, \ldots, g_v$. A group and an object are connected if the object is assigned to the group by one of the models. A group obtained by a classification model links to the node that corresponds to the groundtruth label. We simplify the aforementioned toy example to include the first two classifiers and the first two clustering solutions only, and we show the groups obtained from these four models in Figure 3.1. The group-object bipartite graph is shown in Figure 3.2.

The affinity matrix $A_{n\times v}$ of this graph summarizes the outputs of m models on X:

$$a_{ij} = \begin{cases} 1 & x_i \text{ is assigned to group } j \text{ by a model,} \\ 0 & \text{otherwise.} \end{cases}$$

We try to assign each object into a class based on the consensus among base solutions. For object

x_i, we use indictor variable u_{iz} to indicate its predicted label:

$$u_{iz} = \begin{cases} 1 & \text{the ensemble assigns } x_i \text{ to class } z, \\ 0 & \text{otherwise.} \end{cases}$$

Then $\vec{u}_{i\cdot}$ is a row vector with only one non-zero entry, which is the class that the consensus method assigns x_i to. Putting all the predictions for the data set X together, we get an $n \times c$ matrix U. Optimal integer solutions for an optimization problem are hard to obtain, and we want to have confidence information associated with the final label prediction. Therefore, we replace the constraint that u_{iz} must be 0 or 1 by the weaker constraint that each variable belongs to the interval [0,1]. Now u_{iz} denotes the conditional probability of object x_i belonging to class z. As a nuisance parameter, the conditional probabilities at each group node g_j are also estimated. These conditional probabilities are summarized in two matrices: $U_{n\times c}$ for object nodes and $Q_{v\times c}$ for group nodes. Each entry of the matrices, u_{iz} and q_{jz}, denotes the probability of object x_i and group g_j belonging to class z respectively:

$$u_{iz} = \hat{P}(\delta_{iz} = 1|x_i) \quad \text{and} \quad q_{jz} = \hat{P}(\delta_{jz} = 1|g_j).$$

δ_{iz} or δ_{jz} is an indicator variable, which indicates x_i or g_j belongs to class z if its value is 1. Since the first $s = rc$ groups are obtained from classifiers, they have initial class label estimates denoted by $Y_{v\times c}$ where

$$y_{jz} = \begin{cases} 1 & g_j\text{'s predicted label is } z,\ j = 1, \ldots, s \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Let $k_j = \sum_{z=1}^{c} y_{jz}$. We formulate the consensus agreement as the following optimization problem on the graph:

$$\mathbf{P}: \qquad \min_{Q,U} \quad \varphi(Q,U) = \Big(\sum_{i=1}^{n}\sum_{j=1}^{v} a_{ij}||\vec{u}_{i\cdot} - \vec{q}_{j\cdot}||^2 + \alpha \sum_{j=1}^{v} k_j||\vec{q}_{j\cdot} - \vec{y}_{j\cdot}||^2\Big) \tag{3.2}$$

$$\text{s.t. } \vec{u}_{i\cdot} \geq \vec{0},\ |\vec{u}_{i\cdot}| = 1,\ i = 1:n \tag{3.3}$$

$$\vec{q}_{j\cdot} \geq \vec{0},\ |\vec{q}_{j\cdot}| = 1,\ j = 1:v \tag{3.4}$$

Algorithm 1 BGCM algorithm

```
Input: group-object affinity matrix $A$, initial labeling matrix $Y$; parameters $\alpha$ and $\epsilon$;
Output: consensus matrix $U$;
Algorithm:
  Initialize $U^{(0)}$,$U^{(1)}$ randomly without violating Eq. (3.3)
  Set step number $t \leftarrow 1$
  while $||U^{(t)} - U^{(t-1)}|| > \epsilon$ do
    $t \leftarrow t + 1$
    $Q^{(t)} = (D_v + \alpha K_v)^{-1}(A^T U^{(t-1)} + \alpha K_v Y)$
    $U^{(t)} = D_n^{-1} A Q^{(t)}$
  end while
  return $U^{(t)}$
```

where $||.||$ and $|.|$ denote a vector's L2 and L1 norm respectively. The first term ensures that if an object x_i is assigned to group g_j by one of the models, their conditional probability estimates for the category label must be close. When $j = 1, \ldots, s$, the group node g_j is from a classifier, so $k_j = 1$ and the second term imposes the constraint that group g_j's consensus class label estimate should not deviate much from its initial class label prediction. α is the shadow price payment for violating the constraints. When $j = s+1, \ldots, v$, g_j is a group from an unsupervised model with no such constraints. Thus $k_j = 0$ and the weight of the constraint is 0. Finally, $\vec{u}_{i\cdot}$ and $\vec{q}_{j\cdot}$ are probability vectors, so each component must be greater than or equal to 0, and the sum of the components must be 1. The corresponding integral formulation is shown in Eq. (3.16) in Section 3.4.

We propose to solve this problem using block coordinate descent methods as shown in Algorithm 1. At the t-th iteration, if we fix the value of U, the objective function is a summation of v quadratic components with respect to $\vec{q}_{j\cdot}$. It is strictly convex and $\nabla_{\vec{q}_{j\cdot}} \varphi(Q, U^{(t-1)}) = 0$ gives the unique global minimum of the cost function with respect to $\vec{q}_{j\cdot}$:

$$\vec{q}_{j\cdot}^{\,(t)} = \frac{\sum_{i=1}^{n} a_{ij} \vec{u}_{i\cdot}^{\,(t-1)} + \alpha k_j \vec{y}_{j\cdot}}{\sum_{i=1}^{n} a_{ij} + \alpha k_j} \tag{3.5}$$

Similarly, fixing Q, the unique global minimum with respect to $\vec{u}_{i\cdot}$ is also obtained:

$$\vec{u}_{i\cdot}^{\,(t)} = \frac{\sum_{j=1}^{v} a_{ij} \vec{q}_{j\cdot}^{\,(t)}}{\sum_{j=1}^{v} a_{ij}} \tag{3.6}$$

The update equations in matrix form are given in Algorithm 1. $D_v = \mathrm{diag}\big\{(\sum_{i=1}^{n} a_{ij})\big\}_{v \times v}$ and

Table 3.3: Iterations of Algorithm 1 on an Example

$Q^{(2)}$	$U^{(2)}$	$Q^{(3)}$
(0.6,0.2,0.2)	(0.4667,0.2667,0.2667)	(0.66,0.18,0.16)
(0.1667,0.6667,0.1667)	(0.4667,0.2667,0.2667)	(0.1542,0.7167,0.1292)
(0.1667,0.1667,0.6667)		(0.1236,0.1486,0.7278)
(0.6,0.2,0.2)	(0.3667,0.3667,0.2667)	(0.6583,0.1833,0.1583)
(0.2,0.6,0.2)		(0.1767,0.6417,0.1817)
(0.1111,0.1111,0.7778)	(0.2583,0.4833,0.2583)	(0.0787,0.0787,0.8426)
(0.3333,0.3333,0.3333)		(0.3014,0.3014,0.3972)
(0.3333,0.3333,0.3333)	(0.2583,0.3533,0.3833)	(0.4667,0.2667,0.2667)
(0.3333,0.3333,0.3333)		(0.2917,0.4083,0.3)
(0.3333,0.3333,0.3333)	(0.2361,0.2361,0.5278)	(0.3299,0.3424,0.3278)
(0.3333,0.3333,0.3333)		(0.3625,0.3125,0.325)
(0.3333,0.3333,0.3333)	(0.3583,0.3833,0.2583)	(0.3667,0.3667,0.2667)

$D_n = \text{diag}\{(\sum_{j=1}^{v} a_{ij})\}_{n\times n}$ act as the normalization factors. $K_v = \text{diag}\{(\sum_{z=1}^{c} y_{jz})\}_{v\times v}$ indicates the existence of constraints on the group nodes. During each iteration, the probability estimates at each group node (i.e., Q) combine their initial values Y and the information from the node's neighboring object nodes, then each group node propagates the updated probability estimates back to its neighboring object nodes when updating U. It is straightforward to prove that $(Q^{(t)}, U^{(t)})$ converges to a stationary point of the optimization problem [16].

Example. Table 3.3 shows some intermediate results of the algorithm (with $\alpha = 2$) for the simple example shown in Figure 3.2. Suppose we set each probability vector in $U^{(1)}$ to represent uniform distributions over the classes: (0.3333,0.3333,0.3333). During the first iteration, if a group is obtained from a clustering model, its probabilities are calculated as the average probabilities of the objects that it links to, and thus remains (0.3333,0.3333,0.3333). On the other hand, if a group is from a classifier, we incorporate the prior label information (with a weight α) into the calculation. For example, group g_1 corresponds to the first class and contains objects x_1, x_2 and x_3, and thus its probability vector $\vec{q}_{1\cdot}^{(2)}$ is the weighted average of $\vec{u}_{1\cdot}^{(1)}$, $\vec{u}_{2\cdot}^{(1)}$, $\vec{u}_{3\cdot}^{(1)}$ and $\alpha \cdot (1,0,0)$, which leads to (0.6,0.2,0.2). Similarly, the probabilities of each record ($U^{(2)}$) are calculated as the average probabilities of the groups that it links to. For example, the first object x_1 is adjacent to g_1, g_4, g_8 and g_{10}, and thus $\vec{u}_{1\cdot}^{(2)} = ((0.6, 0.2, 0.2) + (0.6, 0.2, 0.2) + (0.3333, 0.3333, 0.3333) + (0.3333, 0.3333, 0.3333))/4 = (0.4667, 0.2667, 0.2667)$. Such propagation continues until convergence.

3.3 Performance Analysis

In this section, we show that Algorithm 1 gives the optimal solution to the proposed problem with linear convergence rate and also analyze its time complexity.

Convexity of the Problem. We first prove that the optimization problem **P** defined in Eq. (3.2) is a convex program by the following theorem.

Theorem 1. **P** is a convex program.

Proof. Since the constraints of **P** are all linear, we only need to show the objective function of **P**, denoted by $\varphi(Q,U)$, is convex. It can be derived that:

$$\begin{aligned}
\varphi(Q,U) &= \sum_{i=1}^{n}\sum_{j=1}^{v} a_{ij}||\vec{u}_{i\cdot} - \vec{q}_{j\cdot}||^2 + \alpha\sum_{j=1}^{v} k_j||\vec{q}_{j\cdot} - \vec{y}_{j\cdot}||^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{v}\sum_{z=1}^{c} a_{ij}(u_{iz} - q_{jz})^2 + \sum_{j=1}^{v}\sum_{z=1}^{c} \alpha k_j(q_{jz} - y_{jz})^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{v}\sum_{z=1}^{c} a_{ij}(u_{iz} - q_{jz})^2 + \sum_{j=1}^{v}\sum_{z=1}^{c} \alpha k_j q_{jz}^2 + \sum_{j=1}^{v}\sum_{z=1}^{c} \alpha k_j(y_{jz}^2 - 2y_{jz}q_{jz})
\end{aligned} \tag{3.7}$$

Suppose θ is a vector containing all the variables of $\varphi(Q,U)$, i.e., $\theta = (q_{11}, ..., q_{nc}, u_{11}, ..., u_{vc})$. Consider $\varphi(Q,U)$'s standard quadratic form:

$$\varphi(Q,U) = \varphi(\theta) = \theta^T W\theta + b^T\theta + c \tag{3.8}$$

where W, b and c are the coefficient matrix, vector, and scalar of $\varphi(\theta)$ respectively. From Eq. (3.7), we have

$$\theta^T W\theta = \sum_{i=1}^{n}\sum_{j=1}^{v}\sum_{z=1}^{c} a_{ij}(u_{iz} - q_{jz})^2 + \sum_{j=1}^{v}\sum_{z=1}^{c} \alpha k_j q_{jz}^2 \tag{3.9}$$

Note that a_{ij} and αk_j are non-negative for any i and j. Furthermore, we assume that each object receives predictions from at least one model, and thus there exists at least one non-zero entry in each vector. It is obvious that $\theta^T W\theta > 0$ if $\theta \neq 0$. Therefore, the matrix W is strictly positive definite, and thus $\varphi(Q,U)$ is strictly convex. **P** is a convex program. □

For a convex problem, any local minimum is also a global minimum [16]. Therefore, the solution found by Algorithm 1 converges to the global minimum.

Constraints. Note that we do not take into account the constraints (Eq. (3.3) and Eq. (3.4)) when solving **P**. In fact, the constraints will be satisfied when we make proper initialization.

Theorem 2. The solution obtained by Algorithm 1 automatically satisfies the constraints Eq. (3.3) and Eq. (3.4).

Proof. Suppose that the initial value $\vec{u}_{i\cdot}^{(1)}$ satisfies the constraints in Eq. (3.3), namely, $\vec{u}_{i\cdot}^{(1)} \geq \vec{0}$ and $|\vec{u}_{i\cdot}^{(1)}| = 1,\ i = 1, 2, ..., n$. It is obvious that the initial probability vector $\vec{y}_{i\cdot}$ also satisfies the constraints as indicated in its definition (Eq. (3.1)). Now we prove the theorem by induction. Suppose at step $t-1$, the constraints are satisfied. From Eq. (3.5), we can derive that

$$\begin{aligned}|\vec{q}_{j\cdot}^{(t)}| = \sum_{z=1}^{c} q_{jz}^{(t)} &= \frac{\sum_{z=1}^{c}\sum_{i=1}^{n} a_{ij}u_{iz}^{(t-1)} + \sum_{z=1}^{c} \alpha k_j y_{jz}}{\sum_{i=1}^{n} a_{ij} + \alpha k_j} \\ &= \frac{\sum_{i=1}^{n} a_{ij}|\vec{u}_{i\cdot}^{(t-1)}| + \alpha k_j |\vec{y}_{i\cdot}|}{\sum_{i=1}^{n} a_{ij} + \alpha k_j} = \frac{\sum_{i=1}^{n} a_{ij} + \alpha k_j}{\sum_{i=1}^{n} a_{ij} + \alpha k_j} = 1.\end{aligned}$$

In addition, it is clear that $\vec{q}_{j\cdot}^{(t)} \geq \vec{0}$. Thus, Eq. (3.4) is satisfied at the t-th iteration. Similarly, we can show that

$$|\vec{u}_{i\cdot}^{(t)}| = \sum_{z=1}^{c} u_{iz}^{(t)} = \frac{\sum_{z=1}^{c}\sum_{j=1}^{v} a_{ij}q_{jz}^{(t)}}{\sum_{j=1}^{v} a_{ij}} = \frac{\sum_{j=1}^{v} a_{ij}|\vec{q}_{j\cdot}^{(t)}|}{\sum_{j=1}^{v} a_{ij}} = \frac{\sum_{j=1}^{v} a_{ij}}{\sum_{j=1}^{v} a_{ij}} = 1.$$

Again, $\vec{u}_{i\cdot}^{(t)} \geq \vec{0}$, and thus Eq. (3.3) is satisfied at the t-th iteration as well. □

Theorem 2 guarantees that the solution satisfies the constraints of **P** if the initial $U^{(1)}$ satisfies the constraints. Therefore, the solution given by Algorithm 1 is feasible, and we have proved its optimality in Theorem 1.

Time Complexity. It can be seen that at each iteration, the algorithm takes $O(nvc)$ time to compute the probability vectors of groups and objects where n is the number of objects, v is the number of groups and c is the number of classes. The convergence rate of coordinate descent methods is usually linear. For the task of classification, an approximate solution to the optimization problem may suffice. In the experiments reported in Section 3.6, we fix the number of iterations to 40 and get good results. Suppose there are m models and each of the clustering models outputs c clusters. Then $v = mc$ and thus the time complexity of the method is $O(nmc^2)$. As can be seen, the running time is linear with respect to the number of objects and the number of base models, but it is quadratic in the number of classes. For problems with a small number of classes, the proposed method can scale well to large data sets. Scalability experiments shown in Section 3.6 support this claim.

Discussion. We assume that most of the base models are relevant to the classification task, and these models provide complementary expertise, so their consensus represents the best solution. This implicitly assumes that clusters in each clustering solution are correlated with classes. Therefore, although the framework can be applied to the cases with different number of clusters and classes in different solutions, it is reasonable to set the number of clusters equal to the number of classes in usual practice since it approximates such a correlation relationship assumption. If a few models are irrelevant, the results of model combination will not be hurt much. However, we must ensure that most of the models help the classification task. This can be achieved by selecting appropriate data sources by domain experts and choosing the right features for classification or clustering. For example, clustering of product reviews may not be a good model for user sentiment analysis because the clusters are more likely to be formed by product features instead of sentiments. In this case, the model should be learnt using sentiment words only (such as adjectives). Even if we include such a product-oriented clustering model in the consensus combination, the proposed method can compute a low weight for the model if the other classification or clustering models are obtained based on sentiment-oriented features. Then the decisions made by the product-oriented model will not be counted a lot in the final solution.

Another important issue is how the proposed method is related to traditional supervised and unsupervised learning ensemble methods when there are only supervised models (no clustering solutions) or vice versa. Traditional supervised ensemble methods such as bagging and boosting, combine multiple models trained from multiple samples of one training set, and thus they have a different setting from the proposed multiple source knowledge integration. The proposed method is also different from simple majority voting because it actually gives a weighted combination of models. The weights are implicitly computed to simulate model importance such that the models that are more likely to be consistent with the others receive higher weights (encoded in the group node probability vectors). On the other hand, the proposed method seeks an answer to a classification task instead of a clustering partition, so it is different from traditional clustering ensemble techniques. The proposed method requires some label information to guide the label propagation procedure and compute the probability vectors. Therefore, at least one classification model is needed in the collection of base models.

In the proposed bipartite graph, each class/cluster produced by a base model is simply represented as a group node, so it is general enough to cover several more complicated scenarios. For example, when each base model gives a probabilistic label assignment for each object, we can set the weight of the edge between an object and each of its possible groups as the probability of this object belonging to the group. Similarly, the object that has multiple labels to describe its category can link to more than one group in one base solution. We can allow some missing values in the matrix holding all the outputs of the base models, which correspond to the missing edges in the graph. A few missing edges do not affect the consensus maximization procedure.

3.4 Interpretations

In this part, we explain the proposed method from two independent perspectives.

Constrained Embedding. Now we go back to the integral consensus solution, i.e., each object is assigned to exactly one class. So U and Q are indicator matrices. u_{iz} (or q_{jz}) $= 1$ if the ensemble assigns x_i (or g_j) to class z, and 0 otherwise. For group nodes from supervised models, they have been assigned a class label by one of the classifiers, that is, $q_{jz} = y_{jz}$ for $1 \le j \le s$. Because U and Q represent the consensus, we should let group g_j correspond to class z if most of the objects in group g_j correspond to class z in the consensus solution. The optimization is thus:

$$\min_{Q,U} \sum_{j=1}^{v}\sum_{z=1}^{c} \left| q_{jz} - \frac{\sum_{i=1}^{n} a_{ij}u_{iz}}{\sum_{i=1}^{n} a_{ij}} \right| \tag{3.10}$$

$$\text{s.t.} \sum_{z=1}^{c} u_{iz} = 1 \quad \forall i \in \{1,\ldots,n\} \tag{3.11}$$

$$\sum_{z=1}^{c} q_{jz} = 1 \quad \forall j \in \{s+1,\ldots,v\} \tag{3.12}$$

$$u_{iz} \in \{0,1\} \quad q_{jz} \in \{0,1\} \tag{3.13}$$

$$q_{jz} = 1 \quad \forall j \in \{1, \dots, s\} \quad \text{if } g_j\text{'s label is } z \tag{3.14}$$

$$q_{jz} = 0 \quad \forall j \in \{1, \dots, s\} \quad \text{if } g_j\text{'s label is not } z \tag{3.15}$$

Here, the two indicator matrices U and Q can be viewed as embedding $x_1, \dots, x_n$ (object nodes) and $g_1, \dots, g_v$ (group nodes) into a c-dimensional cube. Due to the constraints in Eq. (3.13), $\vec{u}_{i\cdot}$ and $\vec{q}_{j\cdot}$ reside on the boundary of the $(c-1)$-dimensional hyperplane in the cube. $\vec{a}_{\cdot j}$ denotes the objects that group g_j contains. $\vec{q}_{j\cdot}$ can be regarded as the group representative in this new space, and thus it should be close to the group mean: $\frac{\sum_{i=1}^n a_{ij}\vec{u}_{i\cdot}}{\sum_{i=1}^n a_{ij}}$. For the s groups obtained from supervised models, we know their "ideal" embedding, as represented in the constraints in Eq. (3.14) and Eq. (3.15). Note that this formulation is not a clustering procedure. If we regard $\vec{q}_{j\cdot}$ as the j-th cluster center, and $\vec{u}_{i\cdot}$ as the i-th object, we can see that k-means clustering tries to compute the cluster assignment a_{ij} and the cluster center $\vec{q}_{j\cdot}$. However, in our formulation, a_{ij} is fixed, instead, $\vec{u}_{i\cdot}$ is the unknown variable, and thus this formulation simulates an embedding instead of a clustering.

We now relate this problem to the optimization framework discussed in Section 3.2. a_{ij} can only be 0 or 1, and thus Eq. (3.10) just depends on the cases when $a_{ij} = 1$. When $a_{ij} = 1$, regardless of whether q_{jz} is 1 or 0, we have $|q_{jz}\sum_{i=1}^n a_{ij} - \sum_{i=1}^n a_{ij}u_{iz}| = \sum_{i=1}^n |a_{ij}(q_{jz} - u_{iz})|$. Therefore,

$$\begin{aligned}
\sum_{j:a_{ij}=1}\sum_{z=1}^{c}\left|q_{jz} - \frac{\sum_{i=1}^n a_{ij}u_{iz}}{\sum_{i=1}^n a_{ij}}\right| &= \sum_{j:a_{ij}=1}\sum_{z=1}^{c}\frac{|q_{jz}\sum_{i=1}^n a_{ij} - \sum_{i=1}^n a_{ij}u_{iz}|}{\sum_{i=1}^n a_{ij}} \\
&= \sum_{j:a_{ij}=1}\sum_{z=1}^{c}\frac{\sum_{i=1}^n |a_{ij}(q_{jz} - u_{iz})|}{\sum_{i=1}^n a_{ij}}.
\end{aligned}$$

Suppose the groups found by the base models have balanced size, i.e., $\sum_{i=1}^n a_{ij} = \gamma$ where γ is a constant for $\forall j$. Then we can drop γ from the denominator of the objective function:

$$\sum_{j:a_{ij}=1}\sum_{z=1}^{c}\sum_{i=1}^{n}|a_{ij}(q_{jz} - u_{iz})| = \sum_{i=1}^{n}\sum_{j:a_{ij}=1}a_{ij}\sum_{z=1}^{c}|q_{jz} - u_{iz}| = \sum_{i=1}^{n}\sum_{j=1}^{v}a_{ij}\sum_{z=1}^{c}|q_{jz} - u_{iz}|.$$

Therefore, when the classification and clustering models generate balanced groups, the constrained embedding problem in Eq. (3.10) is equivalent to:

$$\min_{Q,U} \sum_{i=1}^{n} \sum_{j=1}^{v} a_{ij} \sum_{z=1}^{c} |q_{jz} - u_{iz}| \tag{3.16}$$

with the same set of constraints from Eq. (3.11) to Eq. (3.15). It is obvious that this is the same as the optimization problem we propose in Section 3.2 with two relaxations: 1) We transform the hard constraints in Eq. (3.15) to soft constraints where the ideal embedding is expressed in the initial labeling matrix Y and the price for violating the constraints is set to α. 2) u_{iz} and q_{jz} are relaxed to have values between 0 and 1, instead of either 0 or 1, and quadratic cost functions replace the L1 norms. They are probability estimates rather than class membership indicators, and we can embed them anywhere on the plane.

With these relaxations, we build connections between the constrained embedding framework as discussed in this section and the one proposed in Section 3.2. Therefore, we can view our proposed method as embedding both object nodes and group nodes into a hyperlane so that object nodes are close to the group nodes that they link to. The constraints are put on the group nodes from supervised models to penalize the embeddings that are far from the “ideal” ones.

Ranking on Consensus Structure. Our method can also be viewed as conducting ranking with respect to each class on the bipartite graph, where group nodes from supervised models act as queries. Suppose we wish to know the probability of group g_j belonging to class 1, which can be regarded as the relevance score of g_j with respect to example queries from class 1. Let $w_j = \sum_{i=1}^{n} a_{ij}$. In Algorithm 1, the relevance scores of all the groups are learnt using the following equation:

$$\vec{q}_{\cdot 1} = (D_v + \alpha K_v)^{-1}(A^T D_n^{-1} A \vec{q}_{\cdot 1} + \alpha K_v \vec{y}_{\cdot 1}) = D_\lambda (D_v^{-1} A^T D_n^{-1} A) \vec{q}_{\cdot 1} + D_{1-\lambda} \vec{y}_{\cdot 1}$$

where the $v \times v$ diagonal matrices D_λ and $D_{1-\lambda}$'s (j, j) entries are $\frac{w_j}{w_j + \alpha k_j}$ and $\frac{\alpha k_j}{w_j + \alpha k_j}$.

Consider collapsing the original bipartite graph into a graph with group nodes only. Then $A^T A$ is its affinity matrix. After normalizing it to be a probability matrix, we have p_{ij} in $P = D_v^{-1} A^T D_n^{-1} A$, which represents the probability of jumping to node j from node i. The groups that are predicted to be in class 1 by one of the supervised models have 1 at the corresponding entries in $\vec{y}_{\cdot 1}$. Therefore these group nodes are "queries", and we wish to rank the group nodes according to their relevance to these queries.

Our ranking model is related to PageRank model [135] in the following aspects: 1) In PageRank, a uniform vector with entries all equal to 1 replaces $\vec{y}_{\cdot 1}$. In our model, we use $\vec{y}_{\cdot 1}$ to show our preference towards the query nodes, so the resulting scores are biased to reflect the relevance regarding class 1. 2) In PageRank, the weights D_λ and $D_{1-\lambda}$ are fixed constants λ and $1-\lambda$, whereas in our model D_λ and $D_{1-\lambda}$ give personalized damping factors. Each group has a damping factor $\lambda_j = \frac{w_j}{w_j+\alpha k_j}$. 3) In PageRank, the value of link-votes are normalized by the number of outlinks at each node, whereas our ranking model does not normalize p_{ij} on its outlinks, and thus it can be viewed as an un-normalized version of personalized PageRank [84, 184]. When each base model generates balanced groups, both λ_j and outlinks at each node become constants, and the proposed method simulates the standard personalized PageRank.

The relevance scores with respect to class 1 for group and object nodes will converge to

$$\vec{q}_{\cdot 1} = (I_v - D_\lambda D_v^{-1} A^T D_n^{-1} A)^{-1} D_{1-\lambda} \vec{y}_{\cdot 1} \quad \vec{u}_{\cdot 1} = (I_n - D_n^{-1} A D_\lambda D_v^{-1} A^T)^{-1} D_n^{-1} A D_{1-\lambda} \vec{y}_{\cdot 1}$$

respectively. I_v and I_n are identity matrices with size $v \times v$ and $n \times n$. The above arguments hold for the other classes as well, and thus each column in U and Q represents the ranking of the nodes with respect to each class. Because each row sums up to 1, each entry in the row is the conditional probability estimate of the node belonging to one of the classes.

3.5 Extensions

In this section, we propose two modified versions of Algorithm 1 to handle a small amount of labeled objects and imbalanced class distributions.

Incorporating Label Information. Thus far, we propose to combine the outputs of supervised and unsupervised models by consensus. When the true labels of the objects are unknown, this is a reliable approach. However, incorporating labels from even a small portion of the objects may greatly refine the final hypothesis. We assume that labels of the first l objects are known, which is encoded in an $n \times c$ matrix F:

$$f_{iz} = \begin{cases} 1 & x_i\text{'s observed label is } z,\ i = 1, \ldots, l, \\ 0 & \text{otherwise.} \end{cases}$$

We modify the objective function in Eq. (3.2) to penalize the deviation of $\vec{u}_{i\cdot}$ from the observed label if the i-th object is labeled:

$$\varphi(Q, U) = \sum_{i=1}^{n} \sum_{j=1}^{v} a_{ij} ||\vec{u}_{i\cdot} - \vec{q}_{j\cdot}||^2 + \alpha \sum_{j=1}^{v} k_j ||\vec{q}_{j\cdot} - \vec{y}_{j\cdot}||^2 + \beta \sum_{i=1}^{n} h_i ||\vec{u}_{i\cdot} - \vec{f}_{i\cdot}||^2 \tag{3.17}$$

where $h_i = \sum_{z=1}^{c} f_{iz}$. When $i = 1, \ldots, l$, $h_i = 1$, we enforce the constraint that object x_i's consensus class label estimate should be close to its observed label with a shadow price β. When $i = l + 1, \ldots, n$, x_i is unlabeled. Therefore, $h_i = 0$ and the constraint term is eliminated from the objective function. To update the conditional probability for the labeled objects, we now incorporate their prior label information:

$$\vec{u}_{i\cdot}^{(t)} = \frac{\sum_{j=1}^{v} a_{ij} \vec{q}_{j\cdot}^{(t)} + \beta h_i \vec{f}_{i\cdot}}{\sum_{j=1}^{v} a_{ij} + \beta h_i} \tag{3.18}$$

In matrix form, this can be written as

$$U^{(t)} = (D_n + \beta H_n)^{-1} (A Q^{(t)} + \beta H_n F) \tag{3.19}$$

with $H_n = \text{diag}\{(\sum_{z=1}^{c} f_{iz})\}_{n \times n}$. Therefore, in the semi-supervised setting, we replace the computation of $U^{(t)}$ in Algorithm 1 by Eq. (3.19). Note that the initial conditional probability of a labeled object is 1 at its observed class label, and 0 at all the others. However, this optimistic estimate will be changed during the updates, with the rationale that the observed labels are just random samples of the true label distribution. Thus we only use the observed labels to bias the updating procedure, instead of totally relying on them.

Example. In Table 3.4, we compare the intermediate results obtained from the unsupervised and semi-supervised versions of the algorithm for the example in Figure 3.2. In the semi-supervised version of the algorithm, we start with uniform probability distributions, and the probability vectors

Table 3.4: Unsupervised vs. Semi-Supervised Algorithms on $U^{(2)}$

Object	Label	Unsupervised	Semi-Supervised
x_1	1	(0.4667,0.2667,0.2667)	(0.8222,0.0889,0.0889)
x_2		(0.4667,0.2667,0.2667)	(0.4667,0.2667,0.2667)
x_3		(0.3667,0.3667,0.2667)	(0.3667,0.3667,0.2667)
x_4	2	(0.2583,0.4833,0.2583)	(0.0861,0.8278,0.0861)
x_5		(0.2583,0.3533,0.3833)	(0.2583,0.3533,0.3833)
x_6	3	(0.2361,0.2361,0.5278)	(0.0787,0.0787,0.8426)
x_7		(0.3583,0.3833,0.2583)	(0.3583,0.3833,0.2583)

of groups ($Q^{(2)}$) are computed in the same way as in the unsupervised version. Therefore, we only show the comparison on $U^{(2)}$ (with $\alpha = 2$ and $\beta = 8$). Suppose we know the class labels of objects x_1 (class 1), x_4 (class 2) and x_6 (class 3), and all the other objects are unlabeled. When calculating the probability vectors of the labeled objects, we now incorporate the label information. For example, the first object x_1 is labeled as class 1, and it belongs to groups g_1, g_4, g_8 and g_{10}. Therefore, $\vec{u}^{(2)}_{1\cdot} = (8 \cdot (1, 0, 0) + (0.6, 0.2, 0.2) + (0.6, 0.2, 0.2)\ (0.3333, 0.3333, 0.3333) + (0.3333, 0.3333, 0.3333))/(8+1+1+1+1) = (0.8222, 0.0889, 0.0889)$. The probability vector (1, 0, 0) is obtained from the label of object x_1. We average the probability vectors of the groups that x_1 is connected to (with weight 1) as well as that of its label (with weight $\beta = 8$). As can be seen, the probability vectors of the labeled objects (x_1, x_4, x_6) are biased towards their labeled classes in the semi-supervised version, and such label information will be propagated to groups and unlabeled objects during later iterations.

Handling Imbalanced Cases. Usually, Algorithm 1 can make accurate predictions on objects with balanced class distributions. However, when the class distribution is imbalanced, there could be some problems. Considering the computation of $\vec{q}_{j\cdot}$ in Eq. (3.5), the probability of the j-th group will be biased towards the majority class. The reason is that we take equal votes from all the objects, and the overwhelming number of objects from the majority class lead to imbalanced votes. In turn, the probability vector of each object $\vec{u}_{i\cdot}$ will be biased towards the majority class when we average the votes from $\vec{q}_{j\cdot}$. Finally, all the objects will be labeled using the majority class.

To solve this problem, we can simply change the matrix A used in the computation of U from

Table 3.5: Un-normalized vs. Normalized Algorithms on $U^{(2)}$

Object	Un-normalized	Normalized
x_1	(0.4667,0.2667,0.2667)	(0.4588,0.2706,0.2706)
x_2	(0.4667,0.2667,0.2667)	(0.4400,0.2800,0.2800)
x_3	(0.3667,0.3667,0.2667)	(0.3538,0.3538,0.2923)
x_4	(0.2583,0.4833,0.2583)	(0.2431,0.5137,0.2431)
x_5	(0.2583,0.3533,0.3833)	(0.2567,0.3367,0.4067)
x_6	(0.2361,0.2361,0.5278)	(0.1975,0.1975,0.6049)
x_7	(0.3583,0.3833,0.2583)	(0.3373,0.4196,0.2431)

the adjacency matrix of the bipartite graph to a normalized matrix B of the same size:

$$b_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \quad \forall i = 1, \ldots, n \quad \forall j = 1, \ldots, v$$

In unsupervised scenarios, the probability of the i-th object being assigned to class z is thus computed as:

$$\vec{u}_{i\cdot}^{(t)} = \frac{\sum_{j=1}^{v} b_{ij} \vec{q}_{j\cdot}^{(t)}}{\sum_{j=1}^{v} b_{ij}}.$$

In other words, we change $U^{(t)} = D_n^{-1} A Q^{(t)}$ to $U^{(t)} = D_n^{-1} B Q^{(t)}$ in Algorithm 1 where D_n is redefined as $\text{diag}\{(\sum_{j=1}^{v} b_{ij})\}_{n \times n}$.

Now, when calculating U, we normalize the vote from the j-th group by this group's number of out-links in the bipartite graph. In the example shown in Figure 3.2, to calculate $\vec{u}_{3\cdot}$ (the conditional probability of object x_3), we average $\vec{q}_{1\cdot}$, $\vec{q}_{5\cdot}$, $\vec{q}_{7\cdot}$ and $\vec{q}_{12\cdot}$ with weights $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{2}$ and 1 respectively. Note that in the un-normalized version, the weights associated with $\vec{q}_{j\cdot}$ in the averaging are all 1. Normally, if the j-th group corresponds to a majority class, the number of its out-links is large because it connects to many objects, and thus its weight in the computation of $\vec{u}_{i\cdot}$ is low. On the other hand, if the j-th group represents a minority class, its weight is high. Therefore, by taking weighted average of the groups' probability vectors, we force the conditional probability of each object to move toward the minority class side. During the iterations, the probability vectors of groups will be influenced as well. In this way, the objects that belong to the minority class will be classified correctly.

Example. In Table 3.5, we compare the results obtained from the un-normalized and normal-

Table 3.6: Data Sets Description

Data	ID	Category Labels	#target	#labeled
Newsgroup	1	comp.graphics comp.os.ms-windows.misc sci.crypt sci.electronics	1408	160
	2	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	1428	160
	3	sci.cypt sci.electronics sci.med sci.space	1413	160
	4	misc.forsale rec.autos rec.motorcycles talk.politics.misc	1324	160
	5	rec.sport.baseball rec.sport.hockey sci.crypt sci.electronics	1424	160
	6	alt.atheism rec.sport.baseball rec.sport.hockey soc.religion.christian	1352	160
Cora	1	Operating_Systems Programming Data_Structures_Algorithms_and_Theory	603	60
	2	Databases Hardware_and_Architecture Networking Human_Computer_Interaction	897	80
	3	Distributed Memory_Management Agents Vision_and_Pattern_Recognition	1368	100
	4	Graphics_and_Virtual_Reality Object_Oriented Planning Robotics Compiler_Design Software_Development	875	100
DBLP	1	Databases Data_Mining Machine_Learning Information_Retrieval	3836	400

ized versions of the algorithm for the simple example shown in Figure 3.2. In the normalized version of the algorithm, we normalize the votes from the nodes by their out-degrees so that the minority ones will be weighted higher during the computation. In this example, since all the objects are linked to exactly four groups, they should be equally weighted. Therefore, the calculation of probability vectors for the groups ($Q^{(2)}$) is the same in both normalized and un-normalized versions. On the other hand, the out-degrees of group nodes are different. For example, object x_6 is linked to four groups: g_3, g_6, g_7, and g_{10}, whose out-degrees are 2, 1, 2 and 4 respectively. Therefore, when calculating the probability vector of x_6, the weights of the four groups are 1/2, 1, 1/2 and 1/4 respectively: $\vec{u}^{(2)}_{6\cdot} = ((0.1667, 0.1667, 0.6667)/2 + (0.1111, 0.1111, 0.7778) + (0.3333, 0.3333, 0.3333)/2 + (0.3333, 0.3333, 0.3333)/4)/(1/2 + 1 + 1/2 + 1/4) = (0.1975, 0.1975, 0.6049)$. As can be seen, the minority group (g_6) has a higher weight in the voting, and thus the probability distribution of object x_6 is biased towards the class g_6 represents. Through the propagation, the probability of an object (a group) belonging to a minority class is thus increased.

3.6 Experiments

We evaluate the proposed algorithms on fifteen classification tasks from four applications. In each task, we have a target set on which we wish to predict class labels. Clustering algorithms are employed on different views of this target set to obtain the grouping results. To construct the classifiers, we apply supervised learning either to data from the same domain or to data from related domains. These classification models are applied to the target set as well. The proposed algorithm generates a consolidated classification solution for the target set based on both classification and clustering results.

3.6.1 Datasets

The details of the tasks are summarized in Table 3.6.

20 Newsgroup categorization. We construct six learning tasks, each of which involves four classes. The objective is to classify newsgroup messages according to topics. We used the version[1] where the newsgroup messages are sorted by date, and separated into training and test sets. The test sets are our target sets. We learn logistic regression [72] and SVM models [32] from the training sets (supervised models M_1 and M_2), and apply these models to the target sets. Meanwhile, we cluster the target sets using K-means and min-cut clustering algorithms (unsupervised models M_3 and M_4) [98].

Cora research paper classification. We aim at classifying a set of research papers into their areas [130]. We extract four target sets, each of which includes papers from three, four or five areas (details can be found in Table 3.6). The training sets contain research papers that are different from those in the target sets. Both training and target sets have two views: the paper abstracts and the paper citations. We apply logistic regression classifiers and K-means clustering algorithms on the two views of the target sets. Therefore, supervised models M_1 and M_2 represent SVM classifiers on abstracts and citations respectively, and the unsupervised model M_3 or M_4 indicates clustering of abstracts or citations.

DBLP data. We retrieve 4,236 authors from DBLP network[2] and try to predict their research areas. The training sets are drawn from a different domain, i.e., the conferences in each research field. There are also two views for both training and target sets: the publication network and the textual content of the publications. The number of papers an author published in the conference can be regarded as a link feature, whereas the pool of titles that an author published is the text feature. Logistic regression and K-means clustering algorithms are used to derive the predictions on the target set. Similar to Cora dataset, supervised models M_1 and M_2 are classification results based on the two views, whereas M_3 and M_4 are unsupervised clustering models on the two views. We manually label the target set for evaluation.

[1]http://people.csail.mit.edu/jrennie/20Newsgroups/
[2]http://www.informatik.uni-trier.de/∼ley/db/

Sentiment data. We use the Multi-Domain Sentiment Dataset[3], which contains product reviews taken from Amazon.com for many product types (domains). We select four domains: books, DVDs, electronics, and housewares as four target sets. To predict categories (positive or negative) of reviews, we train SVM classifiers [32] using the other three domains and obtain three supervised models (M_1, M_2, M_3). We then use three different clustering algorithms (K-means, min-cut, and hierarchical clustering) on the target set and get three unsupervised models (M_4, M_5, M_6). We combine all the models by the consensus maximization method to predict the sentiment orientation of each review in the target set.

3.6.2 Baseline Methods and Evaluation

We denote the proposed method as Bipartite Graph-based Consensus Maximization (**BGCM**), which combines the outputs of the base models. Only clustering ensembles, majority voting methods, and the proposed BGCM algorithm work at the meta output level. In these methods, raw data are discarded, and only prediction results from multiple models are available. However, majority voting cannot be applied when there are clustering models, because the correspondence between clusters and classes is unknown. Therefore, we compare BGCM with two clustering ensemble approaches (**MCLA** [155] and **HBGF** [54]), which ignore the label information from supervised models, regard all the base models as unsupervised clustering, and integrate the outputs of the base models. We combine all the classification models by majority voting to obtain reference labels. With the help of the hungarian method [24], we map the output clusters generated by the clustering ensemble approaches to the reference labels obtained from classification models.

For the clustering algorithms used in the base models, we map their outputs to the best possible class predictions using the groundtruth class labels. Since the true labels are used to do the mapping, it should be able to generate the best accuracy from these unsupervised models. As discussed in Section 3.5, we can incorporate a few labeled objects, which are drawn from the domain of the target set, into the framework and improve accuracy. This improved version of the BGCM algorithm is denoted as **BGCM-L**, and the number of labeled objects used in each task is shown in Table 3.6. On each task, we repeat the experiments 50 times, each of which has randomly

[3]http://www.cs.jhu.edu/∼mdredze/datasets/sentiment/

Table 3.7: Classification Accuracy Comparison on 20 Newsgroup Dataset

Methods	20 Newsgroups					
	1	2	3	4	5	6
M_1	0.7957	0.8836	0.8539	0.8835	0.8751	0.8887
M_2	0.7724	0.8601	0.8127	0.8683	0.8346	0.8571
M_3	0.8044	0.8795	0.8649	0.8975	0.8723	0.9042
M_4	0.7756	0.8563	0.8142	0.8452	0.8602	0.8580
MCLA	0.7602	0.8135	0.8347	0.8655	0.8287	0.8276
HBGF	0.8107	0.9198	0.8585	0.9074	0.8680	0.9005
BGCM	0.8126	0.9058	0.8602	0.9120	0.8855	0.9065
2-L	0.7992	0.9138	0.8507	0.8745	0.8901	0.8921
3-L	0.8157	0.9203	0.8801	0.9087	0.8909	0.9198
BGCM-L	**0.8308**	**0.9209**	**0.8848**	**0.9236**	**0.8998**	**0.9246**
STD	0.0041	0.0033	0.0034	0.0027	0.0041	0.0030

Table 3.8: Classification Accuracy Comparison on Cora and DBLP Datasets

Methods	Cora				DBLP
	1	2	3	4	1
M_1	0.7746	0.8859	0.8735	0.8934	0.9342
M_2	0.7776	0.8586	0.8547	0.8953	0.8762
M_3	0.7951	0.8834	0.8769	0.8912	0.9384
M_4	0.7600	0.8584	0.7832	0.9113	0.7946
MCLA	0.8602	0.8474	0.8824	0.8551	0.8873
HBGF	0.7797	0.9102	0.8513	0.8802	0.9243
BGCM	0.8680	0.9152	0.8945	0.9146	0.9435
2-L	0.8135	0.8765	0.8865	0.9012	0.9017
3-L	0.8647	0.9090	0.9232	0.9158	0.9310
BGCM-L	**0.8928**	**0.9170**	**0.9313**	**0.9279**	**0.9464**
STD	0.0048	0.0039	0.0062	0.0047	0.0029

chosen target and labeled objects, and report the average accuracy. We also show the standard deviation (**STD**) for BGCM-L method. The baselines share very similar standard deviation with the reported one on each task.

3.6.3 Experimental Results

Accuracy. First, we summarized the classification accuracy of all the baselines and the proposed approaches on the target sets of the first eleven tasks in Tables 3.7 and 3.8. The two single classifiers (M_1 and M_2), and the two single clustering models (M_3 and M_4) usually have low accuracy. By combining all the base models, the clustering ensemble approaches (MCLA and HBGF) can improve the performance over each single model. The proposed BGCM method always outperforms the base models, and achieves better or comparable performance compared with the baseline ensembles. By incorporating a small portion (around 10%) of labeled objects, the BGCM-L method further improves the performance. The consistent increase in accuracy can be observed in all the tasks,

Table 3.9: Experimental Results on Sentiment Dataset

Method	Books	DVDs	Electronics	Housewares
M_1	0.7245	0.7555	0.7205	0.7395
M_2	0.6715	0.6790	0.7055	0.7285
M_3	0.6820	0.6915	0.7925	0.8055
M_4	0.5460	0.5070	0.5650	0.5020
M_5	0.5890	0.5175	0.7335	0.7095
M_6	0.5735	0.5160	0.7255	0.7170
MCLA	0.6980	0.7068	0.7664	0.7482
HBGF	0.6601	0.6976	0.8078	0.7694
BGCM	**0.7625**	**0.7770**	**0.8170**	**0.8130**

where the margin between the accuracy of the best single model and that of the BGCM-L method is from 2% to 10%. Even when taking variance into consideration, the results demonstrate the power of consensus maximization in accuracy improvements. Similar patterns can be found from the results on the sentiment dataset in Table 3.9. Although the unsupervised models are less accurate than the supervised models in this case, incorporating them can still benefit the classification task as these models help improve the diversity of the base models.

Sensitivity. As shown in Figures 3.3 (a) and (b), the proposed BGCM-L method is not sensitive to the parameters α and β. To make the plots clear, we just show the performance on the first task of the first three applications. α and β are the shadow prices paid for deviating from the estimated labels of groups and observed labels of objects, so they should be greater than 0. α and β represent the confidence of our belief in the labels of the groups and objects compared with 1. The labels of group nodes are obtained from supervised models and may not be correct. Therefore, a smaller α usually achieves better performance. On the other hand, the labels of objects can be regarded as groundtruth, and thus a larger β is better. In experiments, we find that when α is below 4, and β greater than 4, good performance is achieved. We let $\alpha = 2$ and $\beta = 8$ to produce the experimental results shown in Tables 3.7, 3.8 and 3.9. To see how the performance varies with the amount of labeled data, we fix the target set at 80% of all the objects, and and vary the percentage of labeled objects from 1% to 20%. The results are summarized in Figure 3.3 (c). In general, more labeled objects help the classification task, and the improvements are more visible on Cora data set. When the percentage reaches 10%, BGCM-L's performance becomes stable.

Number of Models. We vary the number of base models incorporated into the consensus

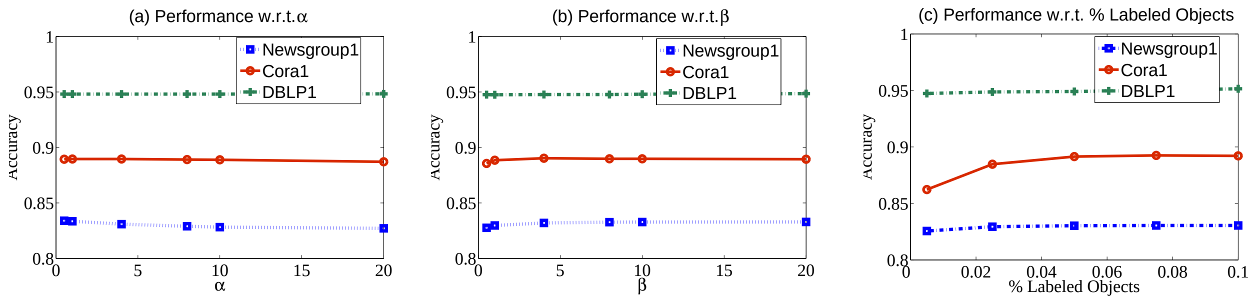


Figure 3.3: Sensitivity Analysis

framework. The BGCM-L method on two models is denoted as **2-L**, where we average the performance of the combined model obtained by randomly choosing one classifier and one clustering algorithm. Similarly, the BGCM-L method on three models is denoted as **3-L**. From Tables 3.7 and 3.8, we can see that BGCM-L method using all the four models outperforms the method incorporating only two or three models. When the base models are independent and each of them obtains reasonable accuracy, combining more models should benefit more because the chance of reducing independent errors increases.

We also ran some simulated experiments on synthetic data with up to 70 models. Suppose we know the true labels of 1000 objects. There are four classes, and each class contains 250 objects. We first discuss how to simulate supervised models. Suppose there are K supervised models, and for each model, we flip some objects' labels from their groundtruth to simulate the outputs for the following two scenarios.

1) The outputs among different models are generated independently, i.e., the models are uncorrelated. To generate the output of a supervised model, we randomly choose $r\%$ of the objects as the objects on which the model makes mistakes. For each of these objects, the model will predict its label to be a randomly-chosen incorrect class.

2) In the second scenario, we want to simulate correlations among the base models. Suppose we have already generated some models. To generate a new model M, we first generate its output independently as described above. Then, we randomly choose an existing model M', and randomly choose 30% objects and force M to make the same predictions on these objects as M'.

We repeat either of the above procedures to generate the outputs of all the K models. Meanwhile, we assume that there are K unsupervised models as well. For unsupervised models, we first

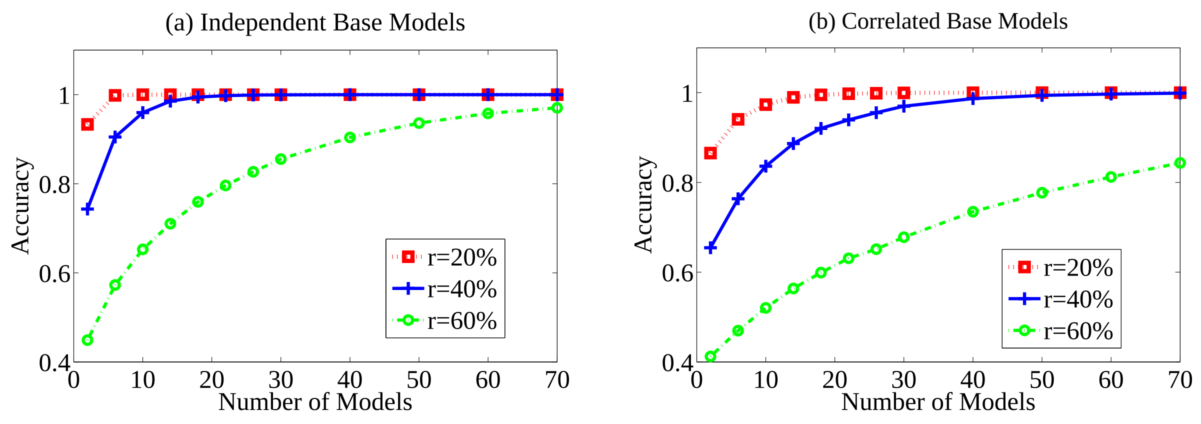


Figure 3.4: Performance Variations w.r.t #Models

generate the label outputs in the same way as described in the supervised model generation. Then we discard the label and randomly assign a cluster ID to each class.

We vary the number of models K from 2 to 70 and for each K, we repeat the above procedure 20 times and average the performance of the proposed method on the 20 data sets. As shown in Figure 3.4, the accuracy of the ensemble obtained by consensus maximization keeps increasing as we incorporate more base models. r is the error rate of the base models. It is obvious that when the base models have higher error rates, a larger number of models are needed to achieve 100% accuracy for the consensus combination. For example, if the base models are independent, and each of which has 20% error rate, we need no more than 10 models in the consensus maximization to reach 100% accuracy. On the other hand, if the error rate of the base models is 60%, we have to combine a lot more models to achieve higher accuracy. By comparing Figures 3.4(a) and 3.4(b), we observe that the accuracy of consensus maximization converges slower when the base models are correlated. For example, when the base models are correlated, and each of them has 20% error rate, we now need to combine around 20 supervised and 20 unsupervised models to achieve an ensemble accuracy of 100%. Therefore, this example provides some guidelines for selecting base models. In general, incorporating more models into consensus maximization will help improve the accuracy, and the more diversified and independent the base models are, the better the combined model performs.

Number of Supervised vs. Unsupervised Models. We also evaluate how the consensus maximization method performs when the ratio between the number of supervised and unsuper-

Table 3.10: Performance Variations w.r.t the Number of Supervised and Unsupervised Models

r	Base Error = 40%			Base Error = 20%		
	S	U	C	S	U	C
1	0.5950	0.6230	0.6333	0.7724	0.8338	0.8463
2	0.5948	0.6119	0.6687	0.7946	0.8298	0.8892
3	0.6518	0.5708	0.6897	0.8418	0.8188	0.8750
4	0.6819	0.5542	0.7260	0.8773	0.8079	0.9153
5	0.7227	0.5622	0.7452	0.8975	0.7956	0.9203
6	0.7308	0.5584	0.7530	0.9224	0.7878	0.9433
7	0.7514	0.5203	0.7652	0.9369	0.7636	0.9430
8	0.7615	0.4838	0.7665	0.9519	0.7491	0.9575
9	0.7869	0.4664	0.8125	0.9557	0.7265	0.9577

vised models changes. We generate synthetic data in the same way as discussed in the experiments on number of models. We set the error rate of base models to be either 40% or 20%. In either way, we generate 10 models and change the number of classification models r from 1 to 9, and accordingly, the number of clustering solutions ranges from 9 to 1. The proposed consensus maximization method combines all the supervised and unsupervised models. As baselines, a majority voting method only combines supervised model outputs, and an unsupervised clustering ensemble approach only combines unsupervised model outputs. As shown in Table 3.10, we compare the proposed method, denoted as "C", with the supervised and unsupervised ensemble approaches, denoted as "S" and "U" respectively. It is clear that the performance of majority voting approach improves, but the accuracy of clustering ensemble approach drops as the number of supervised models increases. In general, incorporating more models into the ensemble help improve the final prediction accuracy in both supervised and unsupervised ensemble approaches. Although the proposed method always combines all the ten models, its performance also improves as the number of supervised models goes up when the total number of models is fixed. This is because the goal of the method is to conduct classification, and more supervised models provide more label information for the task. Therefore, the performance is positively correlated with the number of supervised models. However, since label information is hard to obtain, when we are only given a few supervised models but many unsupervised models, the proposed approach can still gain great benefits from the consensus combination of all these models. As long as most of the base models are relevant to the classification task, and they are drawn from heterogeneous sources, we should combine their complementary expertise.

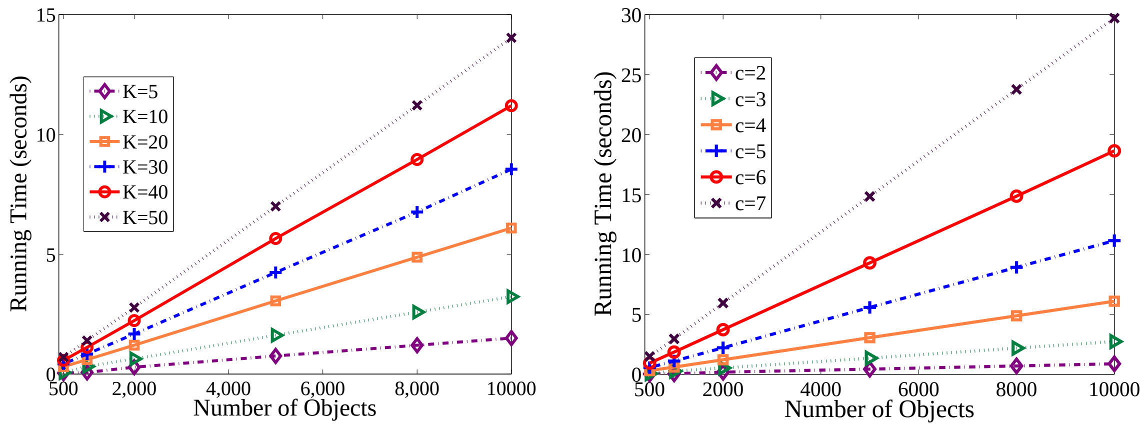


Figure 3.5: Running Time w.r.t. #Models ($2K$) & #Classes (c)

Scalability. As discussed in Section 3.3, the time complexity of the proposed method is quadratic in terms of the number of classes but linear with respect to the number of objects and models. We evaluate the running time of the proposed method on synthetic data sets, which are generated in the same way as in the experiments varying number of models. We generate correlated base models and control the number of classes (c), the number of models (K supervised and K unsupervised models) and the number of objects (n). As shown in Figure 3.5, the running time is linear with respect to the number of objects. The running time increases linearly as the number of models increases, whereas it increases quadratically with respect to the number of classes. Therefore, the results are consistent with our analysis.

3.7 Summary

In this work, we take advantage of the complementary predictive powers of multiple supervised and unsupervised models to derive a consolidated label assignment for a set of objects jointly. We summarize base model outputs in a group-object bipartite graph and maximize the consensus by promoting smoothness of label assignment over the graph and consistency with the initial labeling. The problem is solved by propagating label information between group and object nodes iteratively. We analyze the optimality and time complexity of the proposed solution. The proposed method can be interpreted as conducting an embedding of object and group nodes into a new space. It can also be interpreted as computing an un-normalized personalized PageRank. When a few labeled objects are available, the proposed method can use them to guide the propagation and refine the final hypothesis. If the class distribution is imbalanced, we normalize the out-links of each node in the graph so that the influence of the minority class increases. In the experiments on 20 newsgroup, Cora, DBLP and Sentiment data, the proposed method attains an improvement of between 2% to 10%.

Chapter 4

Consensus Combination for Transfer Learning

In Chapter 3, we presented an effective learning framework to integrate knowledge from multiple heterogeneous information sources with labeled and unlabeled information. Although the proposed framework can be applied to a variety of applications, there are some specific learning scenarios that require more focused solutions. In this chapter and the following chapter, we introduce two specific learning scenarios where the philosophy of model combination can be successfully applied.

In this chapter, we consider an important learning scenario where we wish to transfer labeled information from multiple source domains to a target domain, which is called *transfer learning*. In many applications, we have to borrow labeled information from multiple relevant domains with abundant labeled data (source domains) to classify objects in the domain of interest (target domain). The challenge is that the data from the source domains usually follow different data distributions compared with that in the target domain. To solve this problem, we propose to compute a weighted combination of multiple models derived from source domains where weights are adapted to represent each source domain's predictive power on each target object [62]. Specifically, we map the structures of a model onto the structure of the target domain, and then weight each model locally according to its consistency with the neighborhood structure around each object. Experimental results on text classification, spam filtering and intrusion detection data sets demonstrate significant improvements in classification accuracy gained by the framework. As shown in the transfer learning survey [136], our proposed method outperforms state-of-the-art transfer learning approaches.

4.1 Overview

We are interested in transfer learning scenarios where we learn from one or several training domains and make predictions in a different but related test domain. Such knowledge transfer is possible when the training domain(s) and the test domain have the same set of categories or class labels. We further assume that we are only exposed to some labeled examples from the training domains but do not have any labeled example from the test domain. The study of transfer learning is motivated by the fact that people often exploit knowledge gained from related domains where labeled data are abundant to classify examples in a new domain. Unfortunately, traditional supervised learning techniques usually fail to transfer knowledge in this scenario because it requires the training and the test data to be i.i.d. samples from the same distribution.

There are a few important observations about this problem. We notice that there are usually several classification models available from the training domains. For example, the classifiers can be trained from several relevant domains or built using different learning algorithms on the same domain. Different models usually contain different knowledge and thus have different advantages, due to the inductive bias of the specific learning technique as well as the distributional differences among the training domains. Therefore, different models may be effective at different regions or structures in the new and different test domain, and no single model can perform well in all regions. We refer to these different models as base models. Ideally, we may wish to combine the knowledge from these base models rather than using any single model alone to more effectively transfer the useful knowledge to the new domain. For this task, one would naturally consider model averaging that additively combines the predictions of multiple models. However, the existing model averaging methods in traditional supervised learning usually assign global weights to models, which are either uniform (e.g., in Bagging), or proportional to the training accuracy (e.g., in Boosting), or fixed by favoring certain model (e.g., in single-model classification). Such a global weighting scheme may not perform well in transfer learning because different test examples may favor predictions from different base models. For example, when the base models carry conflicting concepts at a test example, it is essential to select the model that better represents the true target distribution underlying the example. In fact, based on principles of risk minimization, we can derive that there exists a solution to assign per model and per example weights to combine multiple base models to maximize their combined accuracy on the new domain, and the combined accuracy is higher than any single model acting alone. However, it is impossible to dynamically assign the optimal model weights for each example precisely because $P(y|\mathbf{x})$, the true conditional probability of class label y given a test example $\mathbf{x}$, is not known a priori. Past practice of cross-validation based weight assignment is inapplicable since the weights would be assigned based on labels in the given training domain(s) whose $P(y|\mathbf{x})$ could be different from that of the test domain. Therefore our focus is to find an approximation to this optimal local weight assignment for each test example.

We propose a graph-based approach to approximate the optimal model weights where the local weight for a base model is computed by first mapping and then measuring the similarity between the model and the test domain's local structure around the test example. This similarity is measured by comparing neighborhood graphs, and quantified in the weight assignment equation. Intuitively, it favors classifiers whose mapped local structure is similar to the local structure around the test example. For a particular example, if none of the mapped local structures is similar to the original local structure in the target domain, the predicted label will be obtained by voting among its neighbors inside the same local structure of the test set. This strategy ensures that the maximum amount of predictive powers of the labeled information are extracted and transferred to the test domain to make the predictions consistent with its underlying manifold structure.

Our main contributions to the task of transfer learning include the following: (1) We propose a locally weighted ensemble framework to address the transfer learning problem, and demonstrate its superiority over single models in terms of risk minimization when the weights are set optimally. (2) None of the base models is required to be specifically designed for transfer learning, thus providing great flexibility and freedom on what models to use. (3) We propose to approximate the model weights based on the local manifold structures in the test domain, and provide neighborhood graph-based estimation. (4) We provide a prediction adjustment step to propagate labels from nearby examples when all base models are inconsistent with certain test examples.

We evaluate the proposed framework on three real tasks: spam filtering, text categorization, and network intrusion detection. In each task, the test examples come from a different domain than the training set. Our experiment results show that the locally weighted ensemble framework significantly improved the performance over a number of baseline methods on all three data sets,

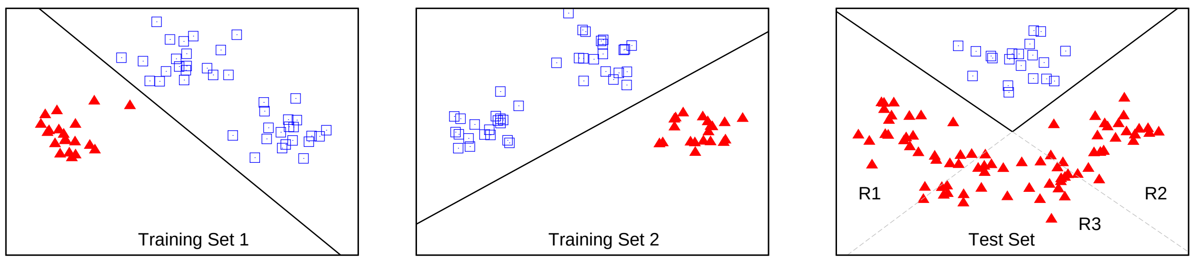


Figure 4.1: A Motivating Example

which shows the effectiveness of the proposed framework for transfer learning.

4.2 Locally Weighted Ensemble

Let us first look at a toy learning problem with two training sets and a test set shown in Figure 4.1. The two training sets have partially conflicting concepts and their decision boundaries are the straight lines. For the test set, however, the optimal decision boundary is the V-shape solid line. As can be seen, the regions R_1 and R_2 are "uncertain," because the two training sets are conflicting there. If we either simply collapse the two data sets and try to train a classifier on the merged examples, or combine the two linear classifiers M_1 and M_2 trained from the training set 1 and set 2 respectively, then those negative examples in R_1 and R_2 will be hard to predict. Those semi-supervised learning algorithms do not work either because they only propagate the labels of the training examples to the unlabeled examples. In this case, there are conflicting labels in R_1 and R_2, causing ambiguous and incorrect information to be propagated. But it is obvious that, if M_1 is used for predicting test examples in R_1 and M_2 used for examples in R_2, then we can label all test examples correctly. Therefore, ideally, one wish to have a "locally weighted" ensemble framework that combines the two models, and weighs M_1 higher at R_1 and M_2 higher at R_2. We also observe that this data set has a property that neighbors along the same "clustering-manifold structure" share the same class labels, which is a commonly-held assumption for reasonable problems. Below, we first introduce a locally weighted ensemble framework with weights dynamically adjusted according to the model behavior at each test example. We then present an effective way of approximating the model weights via local structure mapping around each example. The success of the proposed method on this toy data set is demonstrated in Section 4.4.2.

4.2.1 Optimal Domain Transfer Weights

Let $\mathbf{x}$ be the feature vector and y be the class label where $\mathbf{x}$ and y are drawn from feature space $\mathbf{X}$ and label space Y respectively. For a set of k models $M_1, \ldots, M_k$, the general Bayesian model averaging approach computes the posterior distribution of y as $P(y|\mathbf{x}) = \sum_{i=1}^{k} P(y|\mathbf{x}, D, M_i)P(M_i|D)$, where $P(y|\mathbf{x}, D, M_i) = P(y|\mathbf{x}, M_i)$ is the prediction made by each model and $P(M_i|D)$ is the posterior of model M_i after observing the training set D. However, in transfer learning, since training and test domains are different, we may wish to incorporate information about the test domain and update the model prior for $P(M_i|T)$, where T is the test set. So $P(M_i|D)$ should be replaced by $P(M_i|T)$ in the weighted combination of model predictions. By this replacement, we take the difference between training and test domains into consideration during learning. If the true distribution $P(y|\mathbf{x})$ is known, then for predictions on $\mathbf{x}$, the other examples in the test set are irrelevant to the model performance at $\mathbf{x}$. In other words, the model weight $P(M_i|T)$ is actually $P(M_i|\mathbf{x})$ at $\mathbf{x}$ when $P(y|\mathbf{x})$ is available. Different from traditional ensemble approaches, this locally weighted model averaging method weights individual models according to their local behavior at each test example. The final prediction for $\mathbf{x}$ is:

$$P(y|\mathbf{x}) = \sum_{i=1}^{k} w_{M_i,\mathbf{x}} P(y|\mathbf{x}, M_i), \tag{4.1}$$

where $w_{M_i,\mathbf{x}} = P(M_i|\mathbf{x})$ is the true model weight that is locally adjusted for $\mathbf{x}$ representing the model's effectiveness on the test domain.

The benefits of this locally weighted model averaging approach can be shown as follows. To simplify the problem, we map the label space Y to $\{1, \ldots, c\}$ where c is the number of classes. We then use a $c \times 1$ vector $\mathbf{f}$ to denote the true conditional probability in the test domain where the i-th element is $f_i = P(y = i|\mathbf{x})$. Supervised learning can output a $c \times 1$ vector $\mathbf{h}$ that is close to $\mathbf{f}$ for $\mathbf{x}$. Let $w_i = w_{M_i,\mathbf{x}}$ denote the weight for model M_i at test example $\mathbf{x}$, and let $\mathbf{w}$ denote the $k \times 1$ weight vector. $\mathbf{h}^i$ represents the predictions made by model M_i at $\mathbf{x}$ and is again a $c \times 1$ vector where the j-th element is $h^i_j = P(y = j|\mathbf{x}, M_i)$. $\mathbf{H}$ is used to represent a $c \times k$ matrix with all the model predictions made for $\mathbf{x}$ where the ij entry is model M_i's predicted $P(y = j|\mathbf{x}, M_i)$, i.e., h^i_j. Then the output of the model averaging framework for $\mathbf{x}$ is a vector $\mathbf{h}^e = \mathbf{H}\mathbf{w}$. Note that

$\mathbf{w}$ satisfies the constraints that $w_i \in [0,1]$ and $\sum_{i=1}^{k} w_i = 1$, and thus the output vector $\mathbf{h}^i$ from a single model M_i is a special case of $\mathbf{h}^e$ when $w_i = 1$ and other weights are zero. But we wish to find a weight vector $\mathbf{w}$ which minimizes the distance between $\mathbf{f}$ and $\mathbf{h}^e$. Under squared-error loss, the following objective function should be minimized to obtain the optimal $\mathbf{w}$:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} (\mathbf{f} - \mathbf{H}\mathbf{w})^T(\mathbf{f} - \mathbf{H}\mathbf{w}) + \lambda(\mathbf{w}^T\mathbf{I} - 1), \tag{4.2}$$

where $\mathbf{I}$ is a $k \times 1$ vector of 1 and λ is the regularization term. It is obvious that Eq. (4.2) represents a least-square linear regression problem and the optimal solution is

$$\mathbf{w}^* = (\mathbf{H}^T\mathbf{H})^{-1}(\mathbf{H}^T\mathbf{f} - \frac{1}{2}\lambda\mathbf{I}). \tag{4.3}$$

λ can be further calculated by substituting the above $\mathbf{w}^*$ to the constraint $(\mathbf{w}^*)^T\mathbf{I} = 1$. Usually w_i^* is a value between 0 and 1 so the weight vector of the optimal ensemble is different from that of the single model. Therefore, the error of the model averaging framework on each test example $\mathbf{x}$ will not be greater than that of any single model:

$$(\mathbf{f} - \mathbf{H}\mathbf{w}^*)^T(\mathbf{f} - \mathbf{H}\mathbf{w}^*) \leq (\mathbf{f} - \mathbf{h}^i)^T(\mathbf{f} - \mathbf{h}^i) \quad \forall i \tag{4.4}$$

Thus, for each test example, there is a smaller chance to make a mistake if we combine the predictions from different models using the optimal weight vector. It is important to note that the optimal weight vectors are different for different test examples, so weights should be decided locally.

This locally weighted ensemble framework differs from traditional model averaging methods in the following ways: 1) In transfer learning problems, the traditional methods of assigning model weights based on training set or assigning fixed prior weights are undesirable. Instead, we do not assume that training and test domains follow same distributions but rather focus on the test set when deriving the best model weights to transfer knowledge across domains. 2) Existing work usually weights each model globally, but the proposed method assigns per example weights to each model to identify variations in model performance for different test examples. As discussed, there may not exist one model globally optimal for all the test examples. Usually, different test examples favor different models and therefore the per example weighting scheme is better than the global weighting scheme in terms of classification accuracy.

One challenge is that the optimal per example weight vectors cannot be computed exactly in reality, since the true target vector $\mathbf{f}$ for each test example $\mathbf{x}$ is not known a priori. Importantly however, from its solution in Eq. (4.3), a model will have a higher weight if its prediction on $\mathbf{x}$ is closer to the true $P(y|\mathbf{x})$. In the rest of this chapter, we propose a graph-based approach to approximate the optimal per example weight $w_{M_i,\mathbf{x}}$ under the "clustering-manifold" assumption that $P(\mathbf{x})$ is related to $P(y|\mathbf{x})$. Other approximation heuristics can be developed under this locally weighted framework as long as the weights reasonably approximate the model performance for given test examples.

4.3 Graph-based Weight Estimation

As discussed in Section 4.2.1, the optimal weights can be approximated by assigning a higher weight to a model that produces a more accurate label prediction for $\mathbf{x}$. So the main task is to formulate similarity between the model predictions and the unknown true target function. To achieve this goal, we can model the underlying $P(\mathbf{x})$ from the unlabeled test set in order to infer $P(y|\mathbf{x})$. Specifically, we make a "clustering-manifold" assumption, as commonly held in semi-supervised learning, that $P(y|\mathbf{x})$ is not expected to change much when the marginal density $P(\mathbf{x})$ is high. In other words, the decision boundary should lie in areas where $P(\mathbf{x})$ is low. Under such an assumption, we can compare the difference in $P(y|\mathbf{x})$ between the training and the test data locally with only unlabeled test data. However, probability density estimates are hard to obtain precisely, especially when $\mathbf{x}$ is high-dimensional. Instead, we propose to cluster the test data and assume that the boundaries between the clusters represent the low density areas. As a result, if the local cluster boundaries agree with the classification boundary of M around $\mathbf{x}$, then we assume that $P(y|\mathbf{x}, M)$ is similar to the true $P(y|\mathbf{x})$ around $\mathbf{x}$, and thus the weight for model M ought to be high at $\mathbf{x}$. In the following, we formally give a procedure of computing the weight and illustrate the procedure with an example.

For a test example $\mathbf{x}$ and a base model M to be combined, we first construct two graphs: $G_T = (V, E_T)$ and $G_M = (V, E_M)$. In both graphs, the vertex set V contains all the test examples.

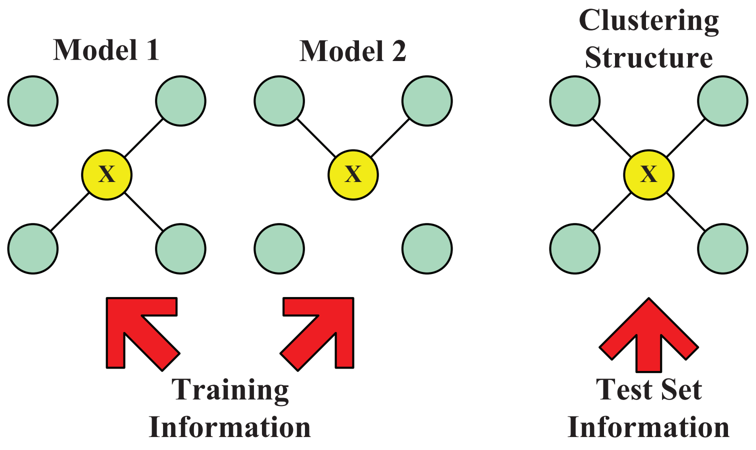


Figure 4.2: Local Neighborhood Graphs around $\mathbf{x}$

For G_M, there is an edge connecting two test examples if and only if the examples are classified into the same class by M. On the other hand, to construct G_T, we cluster the test examples into c' clusters and again, connect two test examples with an edge if and only if the two examples are in the same cluster. Then we can approximate the model weight as the similarity between the local structures around $\mathbf{x}$ in G_T and G_M. Specifically, under the clustering assumption, it is probable that two examples are in the same class if they belong to the same cluster in G_T. So we could use the percentage of common neighbors of $\mathbf{x}$ found in G_M and G_T to approximate the model accuracy on $\mathbf{x}$ and set the weight. Suppose the sets of neighbors for $\mathbf{x}$ in G_M and G_T are V_M and V_T respectively. The model weight at $\mathbf{x}$ is proportional to the similarity of its local structures between G_M and G_T:

$$w_{M,\mathbf{x}} \propto s(G_M, G_T; \mathbf{x}) = \frac{\sum_{v_1 \in V_M} \sum_{v_2 \in V_T} \mathbf{1}\{v_1 = v_2\}}{|V_M| + |V_T|} \tag{4.5}$$

According to its definition, $s(G_M, G_T; \mathbf{x})$ reflects the degree of consistency in labeling the test examples. If $\mathbf{x}$ has similar sets of neighbors in G_M and G_T, it is likely that the model M is consistent with the underlying structure around $\mathbf{x}$. As an example, Figure 4.2 shows the neighborhood graphs of a test example $\mathbf{x}$ constructed from two supervised models and the clustering algorithm on the test set. According to Eq. (4.5), the similarity between model 1 and the clustering structure is 0.75 at $\mathbf{x}$, but that between model 2 and the structure is 0.5. Therefore, for $\mathbf{x}$, model 1's weight will be set higher since it is more consistent with the local structure around $\mathbf{x}$. This is a simple and effective method to compute the similarity.

The weight approximation is based on the clustering assumption which requires that the manifold structure of the data is related to the conditional probability $P(y|\mathbf{x})$. Though this is a reasonable assumption for many problems, it may not always hold. Without knowing $P(y|\mathbf{x})$ a priori, it is impossible to verify the assumption. But this property is usually determined by the nature of the learning tasks. An example where the assumption does not hold is sentiment classification, where the clustering structure of a set of product reviews reveals the topics but may have nothing to do with whether the users like or dislike the product. Therefore, we propose to check the validity of the clustering assumption by evaluating the clustering quality on the training set using purity, entropy or F measure. If the task fails the test, we will ignore the weight approximation step, but simply combine the models using uniform weights. This strategy restricts the use of the graph-based weight estimation only to the cases where the clustering assumption is satisfied on both training and test sets. However, the strict checking criteria could guarantee the high accuracy of the proposed method. For the cases where the clustering assumption does not hold, other techniques need to be explored.

When the condition holds, we compute the per-example model weights based on Eq. (4.5) with a normalization term:

$$w_{M_i,\mathbf{x}} = \frac{s(G_{M_i}, G_T; \mathbf{x})}{\sum_{i=1}^{k} s(G_{M_i}, G_T; \mathbf{x})}, \tag{4.6}$$

where M_i is one of the k models. Then the final prediction of the weighted ensemble E for $\mathbf{x}$ is:

$$P(y|E, \mathbf{x}) = \sum_{i=1}^{k} w_{M_i,\mathbf{x}} P(y|M_i, \mathbf{x}), \tag{4.7}$$

where $P(y|M_i, \mathbf{x})$ is the prediction made by model M_i. Then the predicted label for $\mathbf{x}$ goes to y^* which minimizes the risk:

$$y^* = \arg\min_{y} \int_{y' \in Y} \lambda(y', y) P(y|E, \mathbf{x}) \mathrm{d}y' \tag{4.8}$$

where $\lambda(y', y)$ is the cost incurred when the true class label is y' but the prediction goes to y. With the most commonly used zero-one loss function, $y^* = \arg\max_y P(y|E, \mathbf{x})$.

4.3.1 Local Structure Based Adjustment

The weighting scheme shown in Eq. (4.7) works on the basis that at least some of the models do reasonably well on predicting the label for $\mathbf{x}$. However, if the concepts carried by all the models conflict with the actual concept at $\mathbf{x}$, the similarity measure $s(G_M, G_T; \mathbf{x})$ is expected to be low for each model M. But after the normalization in Eq. (4.6), the locally weighted ensemble framework would still make decisions based on these models for $\mathbf{x}$ and it is probable that the combined output is still in conflict with the true one. In such a scenario, it is reasonable to abandon the labeled information conveyed by the supervised models but rather rely on the local structure around $\mathbf{x}$ only.

Since the similarity measure $s(G_M, G_T; \mathbf{x})$ reflects the degree of consistency between model M's prediction and $\mathbf{x}$'s neighborhood structure, we can use the average $s(G_M, G_T; \mathbf{x})$ over all M to judge whether the labeled information is reliable or not. In fact, $s(G_M, G_T; \mathbf{x})$, representing the average percentage of common neighbors shared by supervised models and clustering results, is within $[0, 1]$. To be exact, when a test example shares the same neighbors in two graphs, their similarity is 1, whereas if no common neighbor is found, it is 0. So for example, if only two models are used and their $s(G_M, G_T; \mathbf{x})$ are both 0.01 at $\mathbf{x}$, then we should avoid normalizing the weights into 0.5 since both models should rather be discarded. Let $s_{avg}(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} s(G_{M_i}, G_T; v)$ be the average similarity between the base models' predictions on $\mathbf{x}$ and the clustering structure around $\mathbf{x}$. Then if $s_{avg}(\mathbf{x}) \geq \delta$, where δ is the threshold, we believe in the prediction obtained from Eq. (4.7); otherwise, we discard all the supervised classifiers and construct an "unsupervised" classifier based on the neighborhood of $\mathbf{x}$.

The "unsupervised" classifier U is not trained on any labeled training set. Its prediction on $\mathbf{x}$ is mainly determined by the neighbors of $\mathbf{x}$ with labels predicted by the combined classifier. Specifically, $P(y|U, \mathbf{x})$ can be decomposed as:

$$P(y|U, \mathbf{x}) = \sum_{C} P(y|U, \mathbf{x} \in C) P(\mathbf{x} \in C|\mathbf{x}). \tag{4.9}$$

Here, C is one of the clusters in the test set. We assume that the cluster membership is deterministic, then $P(\mathbf{x} \in C|\mathbf{x})$ is approximated as follows:

$$P(\mathbf{x} \in C|\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in C \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

Hence, $P(y|U, \mathbf{x})$ is approximately the same as $P(y|U, \mathbf{x} \in C)$ when $\mathbf{x}$ belongs to cluster C. We can further approximate $P(y|U, \mathbf{x} \in C)$ as the average $P(y|E, \mathbf{x})$ for $\mathbf{x} \in C'$ where C' contains test examples which satisfy both $\mathbf{x} \in C$ and $s_{\text{avg}}(\mathbf{x}) \geq \delta$. In other words, only examples that have reliable predictions from the weighted ensemble will count in this procedure. Therefore,

$$P(y|U, \mathbf{x} \in C) \approx \frac{1}{|C'|} \sum_{\mathbf{x} \in C'} P(y|E, \mathbf{x}) \tag{4.11}$$

where $|C'|$ is the size of C'. The above strategy can be simplified if we set $P(y|E, \mathbf{x}) = 1$ when y is the label for $\mathbf{x}$ predicted by E. So $P(y|U, \mathbf{x} \in C)$ can be estimated by a majority vote among examples in C':

$$P(y|U, \mathbf{x} \in C) \approx \frac{P(y, \mathbf{x} \in C'|E)}{P(\mathbf{x} \in C')} \approx \frac{c(y, C'|E)}{|C'|} \tag{4.12}$$

where $c(y, C'|E)$ is the number of examples with label y predicted by ensemble E in C'. So the probability of $\mathbf{x}$ having label y is the percentage of examples in the cluster C' that have y as their class labels, where C' is the cluster that $\mathbf{x}$ belongs to and contains test examples with predicted labels. The final predicted label for $\mathbf{x}$ is determined by Eq. (4.8) with $P(y|E, \mathbf{x})$ replaced by $P(y|U, \mathbf{x})$. If zero-one loss function is applied, the class label for $\mathbf{x}$ whose cluster is C should be the majority label prediction among the test examples which satisfy both $\mathbf{x} \in C$ and $s_{\text{avg}}(\mathbf{x}) \geq \delta$.

4.3.2 Algorithm Description

The framework is summarized in Algorithm 2. We first verify whether the clustering structure is relevant to the classification task by performing clustering on the training set. If the purity of clustering on the training set is below 0.5, we simply combine models using uniform weights. Otherwise, if the clustering quality is satisfactory, in step 2, we construct the neighborhood graphs for both the supervised models and the clustering results. Then in step 3, the weight of each model at each test example is computed, which reflects the consistency of model predictions among the

Algorithm 2 Locally Weighted Ensemble Framework (LWE)

```
Input: (1)A training set $D$ or $k$ training sets $D_1, \dots, D_k$
       (2)$k$ classification models $M_1, \dots, M_k$ ($k > 1$)
       (3)A test set $T$ which comes from a different domain
       but the classification task is the same.
       (4)A threshold $\delta$ and cluster number $c'$.
Output: The set of predicted labels $Y$ for examples in $T$.
Algorithm:

  1. Perform clustering on the training set(s), IF the average purity of clustering is less
     than 0.5, set $w_{M_i,\mathbf{x}} = \frac{1}{k}$ for all $M_i$ and $\mathbf{x}$, and compute the posterior using Eq.(4.7)
     for each $\mathbf{x} \in T$. RETURN.

  2. Group test examples into $c'$ clusters and construct neighborhood graphs based on
     the clustering results and all the $k$ models. Set $T' = \Phi$.

  3. FOR each $\mathbf{x} \in T$,

       • FOR each model $M_i$, compute the model weight $w_{M_i,\mathbf{x}}$ according to Eq.(4.5).
       • IF $s_{\text{avg}}(\mathbf{x}) \geq \delta$, decide $\mathbf{x}$'s label based on the weighted ensemble's output
         $P(y|E, \mathbf{x})$ obtained using Eq.(4.7). ELSE put $\mathbf{x}$ into $T'$.

  4. FOR each $\mathbf{x} \in T'$, predict $\mathbf{x}$'s label from the "unsupervised" classifier $U$, i.e., esti-
     mate $P(y|U, \mathbf{x})$ using Eq.(4.11) or Eq.(4.12). RETURN.
```

test example's neighborhood. We then separate the test examples by checking if its average model weight is greater than a confidence threshold. For those test examples on which cross domain models can make sufficiently accurate predictions, the final label predictions are decided by the locally weighted ensemble. But, for the test examples that the models are not expected to classify correctly, the labels are determined by majority voting among those neighbors with highly confident predictions within the same cluster structure.

4.4 Experiments

In this part, we demonstrate the effectiveness of the locally weighted ensemble framework. The algorithms are evaluated on various data sets covering many application domains. Results show that the proposed framework could combine the predictive powers obtained from multiple sources and gain great improvements in classification accuracy.

4.4.1 Data Sets and Experiment Setup

We conduct experiments on one synthetic and four real data sets, where training and test distributions are different.

Synthetic Data The two training sets and the test set as shown in Figure 4.1 are generated from several Gaussian distributions with the same variance. In each training set, there are 40 positive and 20 negative examples and in the test set, the number of positive and negative examples are 20 and 40 respectively.

Email spam filtering The email spam data set, released by ECML/PKDD 2006 discovery challenge, contains a training set of publicly available messages and three sets of email messages from individual users as test sets. The 4000 labeled examples in the training set and the 2500 test examples for each of the three different users differ in the word distribution. The aim is to design a server-based spam filter learned from public sources and transfer it to individual users.

Document classification The 20 newsgroups data set contains approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups nearly evenly. The Reuters-21758 corpus contains Reuters news articles from 1987. From the two text collections, we generate nine cross-domain learning tasks. Both text collections have a two-level hierarchy so that each learning task involves a top category classification problem but the training and test data are drawn from different sub categories. For example, the goal is to distinguish documents from two top newsgroup categories: rec and talk. So a training set involves documents from "rec.autos," "rec.motorcycles," "talk.politics" and "talk.politics.misc," whereas the test set includes sub-categories "rec.sport.baseball," "rec.sport.hockey," "talk.politics.mideast" and "talk.religions.misc". The strategy is to split the sub-categories among the training and the test sets so that the distributions of the two sets are similar but not exactly the same. The tasks are generated in the same way as in [41] and more details can be found there.

Intrusion detection The KDD cup'99 data set consists of a series of TCP connection records for a local area network. Each example in the data set corresponds to a connection, which is labeled

Table 4.1: Data Sets Description

Task	Data Sets	Training	Test

Email Spam Filtering	User1(U00)
	User2(U01)
	User3(U02)
		Public messages	Each user's emails

20 News-group	Comp vs Sci (C vs S) Rec vs Talk (R vs T) Rec vs Sci (R vs S) Sci vs Talk (S vs T) Comp vs Rec (C vs R) Comp vs Talk (C vs T)	Documents from a set of sub categories	Documents from a different set of sub categories
Reuters	Orgs vs People (O vs Pe) Orgs vs Place (O vs Pl) People vs Place (Pe vs Pl)	Documents from a set of sub categories	Documents from a different set of sub categories
Intrusion Detection	DOS	Probing & R2L Intrusions	DOS Intrusions
	Probing	DOS & R2L Intrusions	Probing Intrusions
	R2L	DOS & Probing Intrusions	R2L Intrusions

as either normal or an attack, with exactly one specific attack type. Some high level features are used to distinguish normal connections from attacks, including host, service and traffic features. In the experiments, we use the 34 continuous features. Attacks fall into four main categories: DOS(denial-of-service), R2L(unauthorized access from a remote machine), U2R(unauthorized access to local superuser privileges), Probing(surveillance and other probing). Since in reality, we usually encounter the problem of detecting the variants of known attacks, it is realistic to have one type of intrusions in the training set but another type in the test set. We create three data sets, each contains a set of randomly selected normal examples and a set of attacks from one category. Since the number of U2R attacks is small, we only use examples from DOS, R2L and Probing categories. Then three cross-domain learning tasks are generated by training from two types of attacks to detect another type of attack. The details of the four real tasks are presented in Table 4.1.

Baseline methods We compare the weighted ensemble framework with different learning algorithms. In particular, since most data sets are high-dimensional, the following commonly used algorithms are appropriate choices: 1) Winnow (**WNN**) from learning package SNoW [28], 2) Logistic Regression (**LR**) implemented in BBR package [72]; and 3) Support Vector Machines (**SVM**) implemented in LibSVM [32]. When we only have a single source domain in the training, three single classifiers are trained using the above learning algorithms and combined according to the proposed weighted ensemble framework. But note that the proposed method is a general framework so that any kind of models could be plugged in and transferred to the test domain. Since semi-supervised learning (transductive learning) is closely related to the problem, we compare the proposed method with Transductive Support Vector Machines (TSVM) implemented in SVM light [95]. Furthermore, in the proposed framework, the two main steps are, predicting labels using weighted classifiers if the classifiers are sufficiently accurate in terms of alignment with clustering structures; and propagating the labels of predicted test examples to the unpredicted ones through the clustering structure. To demonstrate the effectiveness of both steps, we include the following three methods in the comparison: 1) A simple model averaging framework (**SMA**) where all model predictions are combined using uniform weights; 2) The locally weighted ensemble framework without the adjustment step, which simply adopts the weighted prediction for each test example. We call it partial locally weighted ensemble method (**pLWE**); 3) The locally weighted ensemble framework (**LWE**) involving both classifier combination and local structure based adjustment. Note that **SMA** is one of the global ensemble methods where the model weights are set the same for all the test examples. Suppose there are k models, then each model will have a weight $\frac{1}{k}$ at every test example. We use the clustering package CLUTO [98], which is designed for high-dimensional data clustering, to cluster the test set. Again, other clustering algorithms could be used as long as the "clustering" assumption is satisfied.

We compare with a set of different baseline methods on the synthetic and intrusion detection data sets. In each task, we have two source domains for training and the remaining one for the testing. The proposed weighted ensemble methods (pLWE and LWE) are built upon two single models trained from the two source domains using SVM. First, we compare pLWE and LWE with the simple averaging method (SMA) based on the two SVM models. Second, we can choose the training set as 1) one of the two source data sets, or 2) the union of the two source data sets. On the three possible training sets, we study the performance of supervised learning models (SVM) and semi-supervised models (TSVM) and compare them with the proposed methods.

Performance measures To compare the performance of the classification methods, we look at a set of standard evaluation metrics. First, we use classification accuracy, which is simply defined as the percentage of correct predictions among all test examples. Second, under squared loss function, the algorithms can be evaluated using Mean Squared Errors defined as follows: $L = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - \mathcal{P}(+|\mathbf{x}_i))^2$ where $f(\mathbf{x}_i)$ is the output of the classifier, which is the estimated posterior probability of $\mathbf{x}_i$ belonging to positive class, $P(+|\mathbf{x}_i)$ is the true posterior probability and $\{\mathbf{x}_i\}_{i=1}^{n}$ represents the test set. Another measure is used in evaluating the intrusion detection task: the area under ROC curve (AUC), the best of which is 1 corresponding to 100% detection and 0% false alarm. In the experiments, we focus on binary classification, but the framework can be easily applied on multi-class tasks.

4.4.2 Performance Evaluation

In this part, we report the experimental results regarding the effectiveness of the locally weighted ensemble. The results clearly demonstrate that on the transfer learning problems where training and testing data have different distributions, the proposed locally weighted ensemble approach greatly outperforms supervised, semi-supervised single-model algorithms, and a simple averaging ensemble.

Performance Study The results of the toy problem introduced in Figure 4.1 are summarized in Figure 4.3. The results of linear SVM on the training sets from two domains are the top two on the left, denoted as M_1 and M_2. Due to the difference between training and test distributions, both make incorrect predictions at "mirrored" areas. After merging the training sets, the SVM model ("ALL" on top right) still does not work and the constructed hyperplane is obviously a horizontal line. This is due to the fact that there exist conflicting concepts in the merged training set. On the other hand, transductive SVM (TSVM bottom left) trained on merged training sets fails as well since the label propagation is confused by the conflicting training examples. Simple averaging of M_1 and M_2, shown as "SMA" (bottom middle) also makes mistakes in the uncertain areas. However, examples incorrectly classified by these methods are now correctly predicted by the locally weighted ensemble approach (LWE) and the decision boundary matches the V-shape well. To see how this works, first, the clustering algorithm discovers the two clusters above and

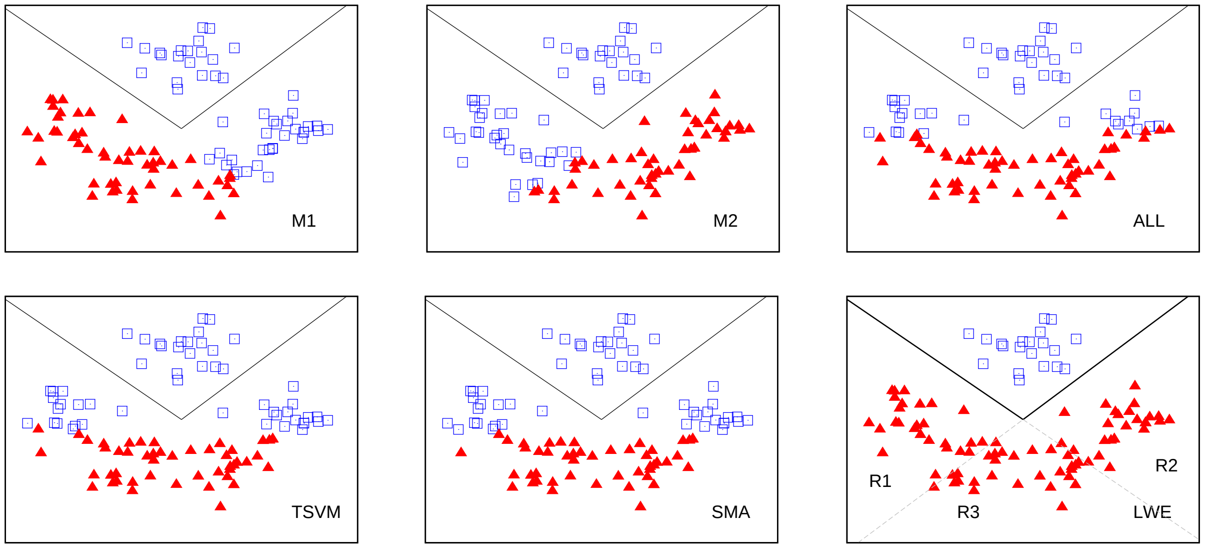


Figure 4.3: Performance on Synthetic Data

below the V-shape. For any example $\mathbf{x} \in R_1$, its neighbors in the cluster contain the examples in all three regions R_1, R_2 and R_3. At the same time, its neighbors predicted by M_1 are those examples $\in R_1$ and R_3. Importantly, its neighbors predicted by M_2 are only examples $\in R_1$. Since there are more common neighbors between the clustering structure and M_1, M_1 will be given higher weight at $\mathbf{x}$. Thus, according to M_1, the examples in R_1 are classified to be negative. Similarly, M_2 will be chosen to predict examples $\in R_2$ as negative. In summary, by weighting the two models locally according to the degree of consistency between models and clusters, the examples at the uncertain areas are predicted correctly.

Results of all the methods on the Email Spam Filtering, 20 Newsgroup and Reuters sets are summarized in Table 4.2 with best results shown in bold font. Refer to Table 4.1 for the details of each task. It is clearly seen that, for all tasks and using any performance measure, the locally weighted ensemble method (LWE) significantly improves the transfer learning performance compared with other baseline methods. We can observe that most of the transfer learning problems are tough due to the unknown discrepancy between the training and the test distributions. The single-model methods (WNN, LR, SVM) usually have poor performance with accuracy around 0.7 and mean squared error greater than 0.1 on most of the tasks. The simple model averaging algorithm using uniform weights can help reduce the expected error compared with single models.

Table 4.2: Performance Comparison on a Series of Data Sets

Accuracy

Methods	Spam Filtering			20 Newsgroup						Reuters		
	U00	U01	U02	C vs S	R vs T	R vs S	S vs T	C vs R	C vs T	O vs Pe	O vs Pl	Pe vs Pl
WNN	0.7680	0.7888	0.8696	0.6554	0.5938	0.7942	0.7557	0.8926	0.9341	0.7058	0.6520	0.5685
LR	0.7060	0.7528	0.8500	0.7349	0.7217	0.7885	0.7904	0.8334	0.9176	0.7355	0.7122	0.5565
SVM	0.6604	0.7288	0.7844	0.7118	0.6824	0.7816	0.7577	0.8156	0.9389	0.6934	0.6998	0.5694
SMA	0.7416	0.8012	0.8768	0.7272	0.6845	0.7980	0.7806	0.8563	0.9348	0.7339	0.7008	0.5685
TSVM	0.8352	0.8512	0.9528	0.7697	0.8995	0.8996	0.8559	0.8964	0.8826	0.7380	0.6989	0.5843
pLWE	0.8584	0.8820	0.9520	0.7872	0.7217	0.8845	0.8330	0.9193	0.9664	0.7694	0.7008	0.5972
LWE	**0.8908**	**0.8844**	**0.9820**	**0.9744**	**0.9923**	**0.9823**	**0.9692**	**0.9816**	**0.9890**	**0.7967**	**0.7304**	**0.6852**

Mean Squared Error

Methods	Spam Filtering			20 Newsgroup						Reuters		
	U00	U01	U02	C vs S	R vs T	R vs S	S vs T	C vs R	C vs T	O vs Pe	O vs Pl	Pe vs Pl
WNN	0.1836	0.1713	0.1003	0.2775	0.2968	0.1575	0.1978	0.0851	0.0525	0.2462	0.3055	0.3774
LR	0.1944	0.1672	0.1013	0.2057	0.2036	0.1567	0.1624	0.1340	0.0613	0.2190	0.2444	0.3900
SVM	0.2374	0.1890	0.1489	0.2140	0.2353	0.1644	0.1826	0.1360	0.0453	0.2217	0.2230	0.2827
SMA	0.1556	0.1337	0.0870	0.2030	0.2183	0.1349	0.1614	0.0979	0.0430	0.1987	0.2318	0.3049
TSVM	0.1428	0.1394	0.0814	0.1749	**0.1080**	0.1128	0.1281	0.1198	0.1061	0.2250	0.2128	0.2688
pLWE	0.1218	0.1012	0.0550	0.1795	0.2027	0.1029	0.1399	0.0699	0.0302	0.1845	0.2333	0.3000
LWE	**0.0988**	**0.1022**	**0.0333**	**0.0965**	0.1409	**0.0384**	**0.0534**	**0.0308**	**0.0140**	**0.1678**	**0.2120**	**0.2091**

However, its performance is not quite satisfactory since they only rely on the labeled information from the source domain and make no efforts in selecting useful information and transferring the knowledge into the test domain. By incorporating the structure information of the test set into learning, the transductive learning approach can beat the supervised learning methods most of the times. But we can see more improvement achieved by using the proposed locally weighted ensemble framework. After the first step of combining classifiers by weighting them judiciously, both accuracy and mean squared error are improved over all the baselines. Then propagating confident predictions along the clustering structure in the test set can significantly boost the performance further. As an example, on the "C vs S" data set in the 20 newsgroup collection, the worst single model only achieves around 66% accuracy whereas the best single model makes correct predictions for 73% of the test examples. The tranductive SVM improves the accuracy to around 77% and LWE outperforms all the other methods by an impressive 97% accuracy. In most of the experiments, the improvement in accuracy after utilizing weighted ensemble is over 10% and up to 30% for some problems. The experimental results on these transfer learning tasks demonstrate the benefits of the empirical approximation of the optimal locally weighted ensemble framework. Both per-example weighting scheme and the adjustment step in the framework can successfully filter out the "harmful" labeled information, and thus help make the most reliable predictions.

Table 4.3 presents the performance of all methods on the three tasks of intrusion detection. Each row corresponds to a learning problem characterized by the test domain and the other two domains act as training, as discussed in Section 4.4.1. Besides the two training domains, a simple

Table 4.3: Performance Comparison on Intrusion Detection Data Set

Accuracy

Intrusions	DOS		Probing		R2L		ALL		SMA	pLWE	LWE
	SVM	TSVM	SVM	TSVM	SVM	TSVM	SVM	TSVM			
DOS	NA	NA	0.9334	0.9352	0.9547	0.9303	0.9294	0.9281	0.9512	0.9609	**0.9623**
Probing	0.8171	0.7820	NA	NA	0.6599	0.8384	0.5808	0.8433	0.5444	0.9627	**0.9636**
R2L	0.5551	0.7602	0.7873	0.8215	NA	NA	0.7615	**0.9036**	0.5360	0.8020	0.8024

AUC

Intrusions	DOS		Probing		R2L		ALL		SMA	pLWE	LWE
	SVM	TSVM	SVM	TSVM	SVM	TSVM	SVM	TSVM			
DOS	NA	NA	0.9774	0.9797	0.9287	0.9188	0.9755	0.9543	0.9854	0.9858	**0.9862**
Probing	0.8877	0.8572	NA	NA	0.5001	0.8982	0.8160	0.8866	0.9745	0.9772	**0.9793**
R2L	0.7114	0.8077	0.9206	0.8727	NA	NA	0.8717	**0.9435**	0.9221	0.9399	0.9418

combination of examples from the two domains (represented as “ALL”) could be another source of training. Based on each training source, we test the performance of SVM and TSVM on the test domain. We also build two single models from each training domain and combine them using uniform weights, which corresponds to SMA. The proposed pLWE and LWE are shown in the last two columns. For the first two learning tasks, it is obvious that the proposed LWE shows dominance for both accuracy and AUC. Especially on the test set of “Probing”, the two training domains seem to be conflicting with each other, thus both the models trained from a union of the two domains and the simple averaging of the two models result in an accuracy around 50% to 60%. LWE achieves 96.36% accuracy by choosing the useful information from the two models. On the last learning task, the algorithm TSVM trained on the combination of training domains wins over the proposed method, which may be due to the fact that one of the single models we are combining has insufficient amount of examples to be relied on. We note that the worst single model's accuracy is around 56% and the simple averaging method even degrades to having 54% accuracy. Based on such weak classifiers, we could still improve the accuracy to 80%.

Parameter Sensitivity There are two important parameters in the proposed algorithm, the number of clusters c' in the test set and the selection threshold δ to filter the predictions with low confidence. The traditional way of setting parameters through cross-validation cannot work when the training and test distributions are different. Again, since the true target function of the test domain is not known, there may not have effective methods to find the optimal values of the parameters. So here, we just give some sensitivity experimental results and state some basic principles in setting the parameters. We choose one cross-domain learning problem from each of the

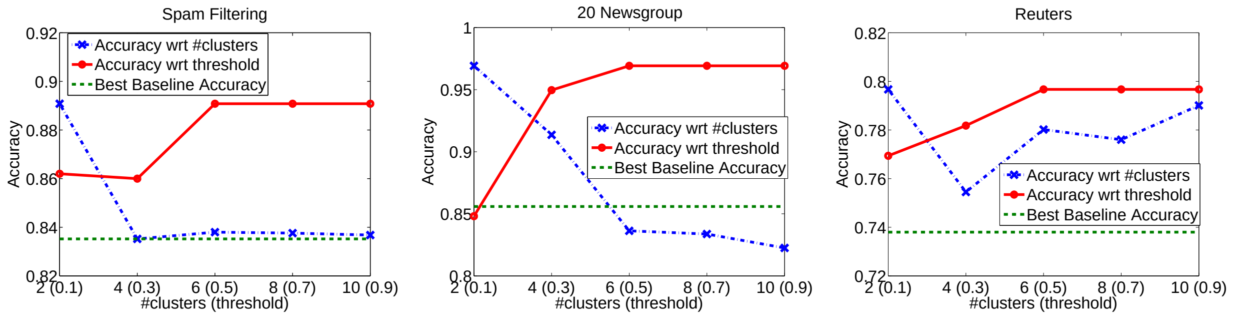


Figure 4.4: Parameter Sensitivity

three data sets: email spam filtering, and 20 newsgroup and Reuters set, and the results are shown in Figure 4.4. We vary c' from 2 to 10 and δ from 0.1 to 0.9, and put both of them on the x-axis. We compare the accuracy of LWE approach when the parameters vary, with that of the best accuracy achieved by the baseline methods. We fix $\delta = 0.7$ when changing c', and let $c' = 2$ when tuning δ. It is clearly seen that when the threshold rises from 0.1 to 0.5, the learning performance on all three sets is gradually improving. After the point of 0.5, the performance maintains stable. This suggests that a low threshold is not desirable since many inaccurate predictions from the supervised models would be used in the adjustment step. Therefore 0.5 up to 1 could be a reasonable range to select the threshold δ. However, the users could choose to lower down or raise the threshold to match their beliefs in the abilities of the supervised models. As for the number of clusters c', the best performance in the experiments is achieved when $c' = 2$. When c' goes up, the over-fitting could occur when the number of examples in each cluster is not sufficient enough to give an accurate estimate of the model weights, and thus we could observe a drop in accuracy. We could also note that in spite of the changes caused by parameter variation, the proposed LWE improves over the best baseline method most of the time.

4.5 Related Work

The problem with different training and test distributions started gaining much attention very recently. When it is assumed that the two distributions differ only in $P(\mathbf{x})$ but not in $P(y|\mathbf{x})$, the problem is referred to as *covariate shift* [150, 87] or *sample selection bias* [178, 50]. The instance weighting approaches [150, 87, 20] try to re-weight each training example with $\frac{P_{\text{test}}(\mathbf{x})}{P_{\text{train}}(\mathbf{x})}$

and maximize the re-weighted log likelihood. Another line of work tries to change the representation of the observation $\mathbf{x}$ hoping that the distributions of the training and the test examples will become very similar after the transformation [14, 144]. [118] transforms the model learned from the training examples into a Bayesian prior to be applied to the learning process on the test domain. The major difference between our work and these studies is that they depend on a single source of information and try to learn a global single model that adapts well to the test set.

Constructing a good ensemble of classifiers has been an active research area in supervised learning [12, 137, 147]. By combining decisions from individual classifiers, ensembles can usually reduce variance and achieve higher accuracy than individual classifiers. Some ensemble methods assign weights locally [5, 91], but such weights are determined based on training data only. There has not been much work on ensemble methods to address the transfer learning problem. In [43, 154], it is assumed that the training and the test examples are generated from a mixture of different models, and the test distribution has different mixture coefficients than the training distribution. In [142], a Dirichlet Process prior is used to couple the parameters of several models from the same parameterized family of distributions. Dai *et al.* [42] extend the boosting method to perform transfer learning. Bennett *et al.* [15] proposed a methodology for building a meta-classifier which combines multiple distinct classifiers through the use of reliability indicators. In [124, 47], a consensus regularization approach has been proposed to enforce multiple classification models agree on the unlabeled data. The proposed weighted ensemble provides a more general framework for transfer learning because 1) the base models can be heterogeneous and can be any generative or discriminative models, and 2) the method does not depend on specific applications and makes no assumption about the form of distributions generating the training or the test data.

Multi-task learning(MTL) [29], which learns several related tasks at the same time with a shared representation, considers single $P(\mathbf{x})$ and multiple output variables, so the basic setting is different from our problem. The "clustering" assumption in our work is exploited in some transfer learning and semi-supervised learning works [41, 187], where clustering structure is utilized in smoothing predictions among neighbors. Our approach differs from these methods by utilizing the assumption in weighting different models locally to combine all sources of labeled information for knowledge transfer.

4.6 Summary

Knowledge transfer across domains with different distributions is an important problem in data mining that has not been fully investigated. In this work, we take advantage of the different predictive powers of several models trained on different domains or using different learning algorithms. We propose a locally weighted ensemble framework to transfer the combined knowledge to a new domain that is different from all the training domains. Importantly, the base models can be constructed by traditional learning algorithms not specifically designed for transfer learning. We analyze the optimality on expected error reduction by utilizing the locally weighted ensemble framework as compared to both single models and globally weighted ensembles. Based on the "clustering" assumption that the local structure of the test set is related to $P(y|\mathbf{x})$, we design an effective weighting scheme to approximate the optimal model weights. This is formulated by comparing the neighborhood graphs of each model with those from clustering. The experimental results on four real transfer learning data sets show that the proposed method improves over each base model 10% to 30% in accuracy and is more accurate than both semi-supervised learning and simple model averaging models. These results indicate that: 1) the locally weighted ensemble could successfully identify the knowledge from each model that is useful to predict in the test domain and transfer such information from all available base models; and 2) the proposed graph-based weight estimation method makes the framework practical by effectively approximating the optimal model weights.

Chapter 5

Consensus Combination for Stream Classification

In this chapter, we address the challenges faced by data stream classification by proposing and analyzing a robust model combination framework. Evolving data streams can be observed in many applications. As there exists distribution evolution or concept drifts, one actually may never know either how or when the distribution changes. In this evolving environment, traditional single-model algorithms that try to match with training distributions will fail. We propose a robust model averaging framework combining multiple supervised models, and demonstrate both formally and empirically that it can reduce generalization errors and outperform single models on stream data [60]. The method is further extended to cope with data streams with imbalanced class distributions [61, 59]. Studies in this chapter draw people's attention to the inevitable concept drifts in data streams, show how the traditional approaches become inapplicable when data distributions evolve continuously, and most importantly, demonstrate the power of ensemble methods in stream data classification. The proposed stream ensemble method has been further extended to handle scarcity of labeled data [129] and novel class detection [127], and has been shown to be effective for malware detection in cyber-security [128].

5.1 Overview

Many real applications, such as network traffic monitoring, credit card fraud detection, and web click stream, generate continuously arriving data, known as data streams [6]. Since classification could help decision making by predicting class labels for given data based on past records, classification on stream data has been extensively studied in recent years, with many interesting algorithms developed [88, 162, 49, 2]. However, there are still some open problems in stream classification as illustrated below.

First, most existing work makes the implicit assumption that the training data and the yet-to-come testing data are always sampled from the "same distribution", and yet this "same distribution" evolves over time. We demonstrate that this may not be true, and one actually may never know either "how" or "when" the distribution changes. Thus, a model that fits well on the observed distribution can have unsatisfactory accuracy on the incoming data. Practically, one can just assume the bare minimum that learning from observed data is better than both random guessing and always predicting exactly the same class label.

Another important issue is that existing stream classification algorithms typically evaluate their performance on data streams with balanced class distribution. It is known that many inductive learning methods that have good performance on balanced data would perform poorly on skewed data sets. In fact, skewed distribution can be seen in many data stream applications. In these cases, the positive instances are much less popular than negative instances. For example, the online credit card fraud rate of US is just 2% in 2006. On the other hand, the loss functions associated with classes are also unbalanced. The cost of misclassifying a credit card fraud as normal will impose thousands of dollars loss on the bank. The deficiency in inductive learning methods on skewed data has been addressed by many people [169, 34, 11]. Inductive learner's goal is to minimize classification error rate, therefore, it completely ignores the small number of positive examples and predicts every example as negative. This is definitely undesirable.

In light of these challenges, we first provide a systematic analysis on stream classification problems in general. We show that the commonly held "shared distribution assumption" may not be appropriate, and stream classification algorithms ought to consider situations where training and testing distributions are different. We suggest a relaxed and realistic assumption as follows and demonstrate the robustness of a model averaging and simple voting-based framework for data streams. We further extends the framework to handle a more challenging situation in stream mining, where the class distributions in the data are skewed. Our main contributions are as follows:

1. We demonstrate that assuming training and testing data follow the same distribution, as commonly held by much existing work, is inappropriate for practical streaming systems. Contrary to common practice, in order to design robust and effective stream mining algorithms against changes, an appropriate methodology is not to overly match the training distribution, such as

by weighted voting or weighed averaging where the weights are assigned according to training distribution.

2. We propose to use both model averaging of conditional probability estimators and simple voting of class labels as a robust framework "against change" and argue that weighted averaging/voting are inappropriate. We demonstrate both formally and empirically such a framework can reduce expected errors and give the best performance on average when the test data does not follow the same distribution as the training data.

3. We adapt the general stream classification framework to classify data streams with skewed class distribution. We employ both sampling and ensemble techniques in the algorithm and show their strengths theoretically and experimentally. The results clearly indicate that our proposed method generates reliable probability estimates and significantly reduces the classification error on the minority class.

The rest of this chapter is organized as follows. Section 5.2 discusses the problems of the assumptions held by existing stream classification algorithms and introduces a realistic assumption. In Section 5.3, we introduce a robust ensemble approach for mining concept-drifting data streams and demonstrate its advantages through theoretical analysis. We present the adaptation of the framework in classifying skewed data streams in Section 5.4. Experimental results on the ensemble approach are given in Section 5.5. Finally, related work and summary of this chapter are presented in Section 5.6 and Section 5.7.

5.2 Appropriate Assumptions to Mine Data Streams

Classification on stream data has been extensively studied in recent years with many important algorithms developed. Much of the previous work focuses on how to effectively update the classification model when stream data flows in [2, 88, 103]. The old examples can be either thrown away after some period of time or smoothly faded out by decreasing their weights as time elapses. Alternatively, other researchers explore some sophisticated methods to select old examples to help train a better model rather than just using the most recent data alone [170, 49, 162, 105, 146]. These algorithms select either old examples or old models with respect to how well they match the current data. Hence, they also implicitly make the assumption that the current training distribution is considerably close to the unknown distribution that produces future data. Among these methods, the weighted ensemble approaches [49, 105, 146, 162] were demonstrated to be highly accurate, when the "stationary distribution assumption" holds true. Formally, we denote the feature vector and class label as $\mathbf{x}$ and y respectively. Data stream could be defined as an infinite sequence of $(\mathbf{x}_i, y_i)$. Training set D and test set T are two sets of sequentially adjacent examples drawn from the data stream. The labels in T are not known during classification process and will only be provided after some period of time. The assumption held by existing algorithms is stated as follows:

Assumption 1 (Shared Distribution - Stationary Distribution)**.** *Training D and test data T are assumed to be generated by the same distribution $P(\mathbf{x}, y) = P(y|\mathbf{x}) \cdot P(\mathbf{x})$ no matter how $P(\mathbf{x}, y)$ evolves as time elapses.*

Given this assumption, one would ask: "what is the difference between stream mining and traditional mining problems?" The most significant difference from traditional "static" learning scenarios is that this shared distribution between training and testing data (abbreviated as "shared distribution" in the rest of this chapter) evolves from time to time in three different ways: (1) *feature changes*, i.e., the changes of the probability $P(\mathbf{x})$ to encounter an example with feature vector $\mathbf{x}$; (2) *conditional changes*, i.e., the changes of the conditional probability $P(y|\mathbf{x})$ to assign class label y to feature vector $\mathbf{x}$; and (3) *dual changes*, i.e., the changes in both $P(\mathbf{x})$ and $P(y|\mathbf{x})$. An illustration with a real-world intrusion dataset can be found in the following discussions.

Under the "shared distribution assumption", the fundamental problems that previous works on stream mining focus on are mainly the following areas: 1) How often the shared distribution changes? It could be continuous or periodical, and fast or slow; 2) How much data is collected to mine the "shared distribution"? It could be sufficient, insufficient or "just don't know"; 3) What is this "shared distribution"? It could be balanced or skewed, binary or multi-class, and etc.; 4) How the shared distribution evolves? There could be conditional change, feature change, or dual change; and 5) How to detect the changes in shared distribution? Some methods do not detect them at all and always keep the models up-to-date whereas others only trigger model reconstruction if a change is suspected. Obviously, the validity of some of these problems relies on the "shared

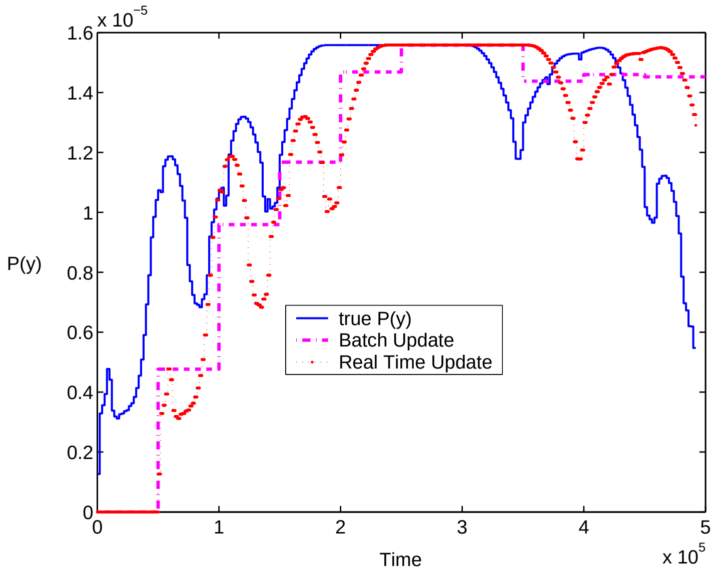


Figure 5.1: Evolution of $P(y)$

distribution assumption", which we challenge below. Interestingly, given "stationary distribution assumption", stream learning would still be effectively the same as traditional learning if the set of training examples collected to mine the "shared distribution" is sufficiently large so that additional examples cannot construct a more accurate model [49].

Realistic Assumption The implicitly held assumption (Assumption 1) may not always be true for data streams. As an example, let us consider the KDDCUP'99 "intrusion detection" dataset that is widely used in the stream mining literature. We plot the evolution on the percentage of intrusions using "averaged shifted histogram (ASH)" in Figure 5.1. The true probability $P(y)$ to encounter an intrusion is shown in thick solid line. Obviously, $P(y)$ is very volatile. As time elapses, $P(y)$ continues to change and fluctuate. At some period, the change is more significant than others. Except for the flat area between time stamps 2×10^5 and 3×10^5, $P(y)$ from the past is always different from that of the future examples. Under "shared distribution" assumption, the training distribution ought to be accurately modeled as the ultimate target. However, it may not precisely match future testing distribution due to continuous change.

The fluctuation in $P(y)$ comes from changes in $P(y|\mathbf{x})$ or $P(\mathbf{x})$. Let $+$ denote intrusions. By definition, $P(y=+) = \frac{\sum P(\mathbf{x},y=+)}{\sum P(\mathbf{x})}$ and $\sum P(\mathbf{x})$ is fixed for a given period, then $P(y) \propto P(\mathbf{x},y) = P(y|\mathbf{x})\cdot P(\mathbf{x})$. Thus, the change in $P(y)$ has to come from $P(y|\mathbf{x})$, or $P(\mathbf{x})$, or possibly both $P(y|\mathbf{x})$

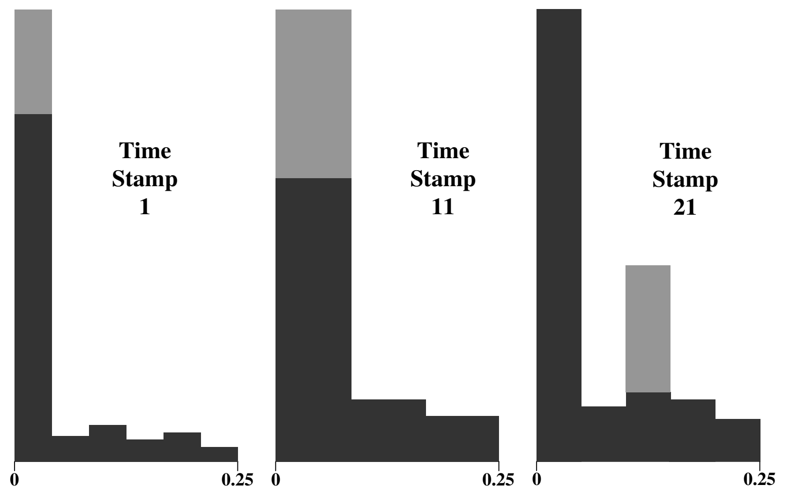


Figure 5.2: Evolution of $P(\mathbf{x})$ and $P(y|\mathbf{x})$

and $P(\mathbf{x})$. Unless the dataset is synthesized, one normally does not know which of these three cases is true, either before or after mining. Because of this, a model constructed from the training data may not be highly accurate on the incoming data. This can particularly be an issue if the changes are attributed to conditional probability $P(y|\mathbf{x})$. As follows, we illustrate how $P(\mathbf{x})$ and $P(y|\mathbf{x})$ change using the same intrusion detection example.

Figure 5.2 shows the histograms of the percentage of intrusions and normal connections given the feature 'srv_diff_host_rate' in three different time periods, where gray represents intrusions and black indicates normal connections. The range of this feature, or the percentage of connections to different hosts, remains within [0,1]. Due to the space limit, we only show the histograms between 0 and 0.25. Most bars between 0.25 and 1 have heights close to 0 and do not reveal much useful information. It is obvious that the distribution of this feature, or visually the relative height of each bar in the histogram representing the percentage of connections, is different among these three time periods. This obviously indicates the change in $P(\mathbf{x})$ as data flows in. In addition, the probability distribution to observe intrusions given this feature is quite different among these three periods. For example, in the first time period, $P(y = +|x \in [0.095, 0.105]) = 0$ but it later jumps to around 0.7 at the last time stamp. In the following, we will discuss how the "shared distribution" assumption affects learning when the actual data evolves in the manner described above. It is worth noting that some stream mining algorithms [170, 103, 49, 162, 105, 176, 146] discuss about the concept drifts in streams and recognize the changes in the distribution that generates the data. However, they still make some assumptions about the forms of concept drifts. For example, most of them assume that the most recent training data is drawn from the distribution which is considerably close to that generates the test data [103, 49, 162, 105, 146].

Depending on when labeled training data becomes available, existing stream classification algorithms belong to two main categories. The first group [2, 88] updates the training distribution as soon as labeled example becomes available and flows in, and at the same time, obsolete examples are either discarded or "weighted" out. Under the "shared distribution" assumption, such method obviously assumes that the distribution of the next moment is the same as those observed data in memory. Visually, it assumes a "shifted" or "delayed" $P(y)$ as the distribution of the future, as shown by the "Real Time Update" curve in Figure 5.1. To be precise, when either the number of examples kept in memory is not sufficiently large or the fading weights are not set properly, $P(y)$ may not only be shifted but also carry a "different shape" from the plot constructed by average shifted histogram. The second family of stream classification algorithms [49, 162, 105, 146] normally receives labeled data in "chunks", and assumes that the most recent chunk is the closest to the future distribution. Thus, they concentrate on learning from the most recent data accurately as well as some old examples that are similar to the current distribution. Due to the changing $P(y)$, we observe both "shift" and "flattening" of the assumed future distribution, shown in the "Batch Update" curve in Figure 5.1. "Flattening" is due to chunking and is hard to avoid since labeled data may arrive in chunks. As a summary, for both families of methods, "shifting" is not desirable and ought to be resolved.

In fact, "shift" or "delay" is inevitable under the "shared distribution assumption", since the culprit is the assumption itself: the future data is not known and can change in different ways from the current data, but they are implicitly assumed to be the same. In order to overcome the "delaying" problem, the main question is how one should judiciously use what is known in order to optimally match the unknown future, with the least surprise and disappointment. Existing algorithms have obviously taken the road to accurately match the training distribution with the hope that it will perform well on the future data. However, from the above example as well as detailed experiments on this example in Section 5.5, they could perform poorly when the future is quite different from the current. By this token, we could see that the commonly held "shared distribu-

- $\mathbf{x}$ is feature vector from feature space X and $P(\mathbf{x})$ is the probability distribution of feature vectors.
- y is the class label from space Y and $P(y)$ is the prior class probability.
- $P(\mathbf{x}, y)$ is the joint probability of having feature vector $\mathbf{x}$ and class label y, and $P(y|\mathbf{x})$ is the conditional probability for $\mathbf{x}$ to have label y.
- Stream data is an infinite sequence of $X - Y$ pairs, $\{(\mathbf{x}_i, y_i)\}$ where the value of y_i is known after a certain time period.
- Since $P(\mathbf{x}, y)$ is evolving in streams, we use $P_t(\mathbf{x}, y)$ to represent the joint distribution over $X - Y$ space at time t.
- Training set D and test set T contain sequentially adjacent examples drawn from the stream data. The true values of y_i in T is not known at the time of learning.
- Training set D is drawn from distribution $P_a(\mathbf{x}, y)$, and test set T is drawn from $P_e(\mathbf{x}, y)$. $a < e$, and $P_a(\mathbf{x}, y)$ and $P_e(\mathbf{x}, y)$ are different.
- $P_a(\mathbf{x}, y)$ and $P_e(\mathbf{x}, y)$ are similar in the sense that the model trained on D and evaluated on T is more accurate than random guessing and fixed prediction.

Figure 5.3: Notations and Assumptions

tion assumption" may not be appropriate, and stream classification algorithms ought to consider situations where training and testing distributions are different. Thus, we take this difference into consideration and suggest a relaxed and realistic assumption as follows:

Assumption 2 (Learnable Assumption)**.** *The training and testing distributions are similar to the degree that the model trained from the training set D has higher accuracy on the test set T than both random guessing and predicting the same class label.*

The core of this new assumption is that it does not assume to know any exact relationship between current training and future test distribution, but simply assume that they are similar in the sense that learning is still useful. As commonly understood, this is the bare minimum for learning. It should be noted that this assumption is made concerning the inductive learning problem. Mining data streams from other perspectives, such as clustering, association mining, may require other appropriate assumptions. All the notations and assumptions we made are summarized in Figure 5.3. With the relaxed assumption, we first elaborate the idea that one should only match the training distribution to a certain degree, then we shall provide a straightforward framework that can maximize the chance for models to succeed on future data with different distributions.

5.3 A Robust and Extensible Ensemble Framework

In Section 5.2, we illustrate that when learning from stream data, it is unlikely that training and testing data always come from the same distribution. This phenomenon hurts existing algorithms that are based upon such an assumption. Some stream mining work has investigated the change detection problem [101] or utilized the concept change in model construction [176]. However, since there are only unlabeled examples available in the test data set, the "change detection" could at most detect feature change. It is rather difficult to detect the change in $P(y|\mathbf{x})$ before class labels are given. The moral of the relaxed assumption (Assumption 2) ought to be understood in the way that "strong assumptions are no good for stream mining". To carry this understanding one step further, any single learning method on data streams also makes assumptions one way or the other on how to match the training distribution effectively and still perform well on testing distribution, and these assumptions can also fail for a continuously changing data stream. Instead, we use a naive model averaging based approach that does not depend specifically on any single technique but combines multiple techniques wherever and whenever available. Formally, suppose k models $\{M_1, M_2, \ldots, M_k\}$ are trained (e.g. using different learning algorithms) and each of them outputs an estimated posterior probability $P(y|\mathbf{x}, M_i)$ for each test example $\mathbf{x}$. We use simple averaging to combine the probability outputs, thus $f^A(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} P(y|\mathbf{x}, M_i)$, and its optimality is discussed below.

Performance Guarantee As described above, we generate k models and each model M_i outputs an estimated probability $P(y|\mathbf{x}, M_i)$ for $\mathbf{x}$. For the sake of simplicity, we use M to denote any of the k models M_i and use Θ_M to represent the collection of the k models. Then any base model M's expected mean squared error is the difference between its predicted probability and the true probability integrated over all test examples:

$$\begin{aligned} Err^M &= \sum_{(\mathbf{x},y)\in T} P(\mathbf{x}, y)(P(y|\mathbf{x}) - P(y|\mathbf{x}, M))^2 \\ &= E_{P(\mathbf{x},y)}[P(y|\mathbf{x})^2 - 2P(y|\mathbf{x})P(y|\mathbf{x}, M) + P(y|\mathbf{x}, M)^2] \end{aligned}$$

Suppose each model M has probability $P(M)$ on the test set, then the expected error incurred by randomly choosing a base streaming model to do prediction is the above error Err^M integrated over all models:

$$
\begin{aligned}
Err^S &= \sum_{M \in \Theta_M} \sum_{(\mathbf{x},y) \in T} P(\mathbf{x}, y)(P(y|\mathbf{x}) - P(y|\mathbf{x}, M))^2 \\
&= E_{P(M),P(\mathbf{x},y)}[P(y|\mathbf{x})^2 - 2P(y|\mathbf{x})P(y|\mathbf{x}, M) + P(y|\mathbf{x}, M)^2]
\end{aligned}
$$

It should be noted that the above equation only evaluates the general performance of base streaming models, but the predictions of test examples are not averaged. Now, we come to the analysis of ensemble where the predictions are averaged. As introduced before, we make the following "model averaging" prediction: $f^A(\mathbf{x}) = E_{P(M)}[P(y|\mathbf{x}, M)]$. Then the expected error of this ensemble should be the error integrated over the universe of test examples:

$$
\begin{aligned}
Err^A =& \sum_{(\mathbf{x},y) \in T} P(\mathbf{x}, y)(P(y|\mathbf{x}) - E_{P(M)}[P(y|\mathbf{x}, M)])^2 \\
=& E_{P(\mathbf{x},y)}[P(y|\mathbf{x})^2 - 2P(y|\mathbf{x})E_{P(M)}[P(y|\mathbf{x}, M)] + E_{P(M)}[P(y|\mathbf{x}, M)]^2] \\
\leq& E_{P(\mathbf{x},y)}[P(y|\mathbf{x})^2 - 2P(y|\mathbf{x})E_{P(M)}[P(y|\mathbf{x}, M)] + E_{P(M)}[P(y|\mathbf{x}, M)^2]]
\end{aligned}
$$

The inequality holds since $E[f(x))]^2 \leq E(f(x)^2]$, i.e., $E_{P(M)}[P(y|\mathbf{x}, M)]^2 \leq E_{P(M)}[P(y|\mathbf{x}, M)^2]$. Therefore, $Err^A \leq Err^M$, i.e., probability averaging of multiple models is superior to any base streaming model chosen at random with respect to reduction in expected errors on all possible examples.

We are not claiming that model averaging is more accurate than any single model at any given time. As a simple illustration, Figure 5.4 shows the errors of three models at time A and time B. At a specific time stamp, a single model M that fits current distribution well could have much better performance on test data than other models, e.g., M2 at time A and M1 at time B. At this same time stamp, the probability averaging of three models (shown as AP) may not necessarily be more accurate than using a specific model. However, in stream learning problems, it is hard to find a single model that works well on all possible training-test pairs drawn independently from continuously changing distributions. Since it is unknown which single model could be optimal at

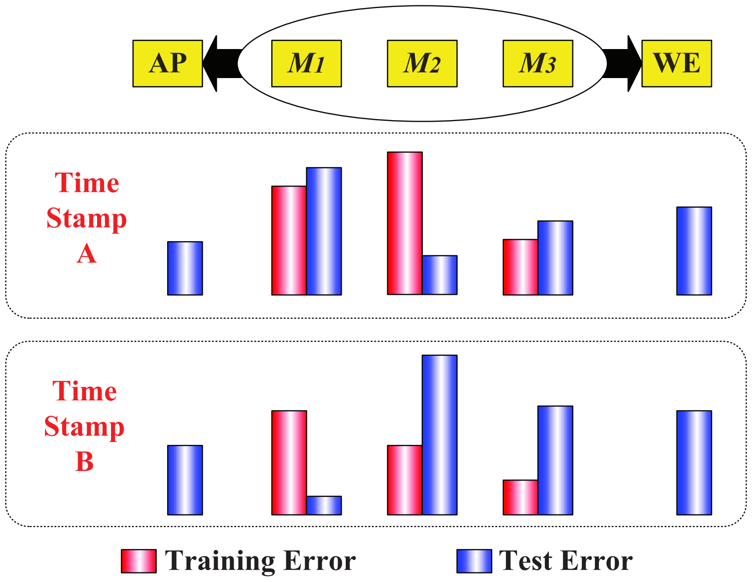


Figure 5.4: Error Comparison between Our Approach and Baselines

each and every time stamp, the current practice is to select a method and hope it will perform the best at any time stamp. However, this could be risky. In the above example, the most accurate model M2 at time stamp A turns out to be the least accurate at time stamp B. On the other hand, the model averaging approach could reduce the probability of surprises and guarantee the most reliable performance. The above analysis formally proves the expected error incurred by randomly choosing a single model is greater than model averaging. Therefore, unless we know exactly which model is always the best, unrealistic in a constantly changing stream environment, we could expect model averaging to have the best expected performance.

Optimality of Uniform Weights The next question is how to decide $P(M)$, or the probability of model M being optimal. The simplest way is to set $P(M^*) = 1$ where M^* is the most accurate model and set other model's probability as 0. This is one of the common practice adopted by some stream mining algorithms where the model itself is fixed but its parameters are re-estimated as labeled data flows in. As discussed above, the expected performance of a single model could be low, when the distribution is continuously evolving. Another more sophisticated approach is introduced in [162], where each model is assigned a weight that is reversely proportional to its error estimated using training data. That is to say, $P(M)$ is higher if the model M incurs less errors when cross-validated using the training data. This weighting scheme is problematic because: 1) the training examples may be insufficient to reflect the true accuracy of model M, thus the weights may not represent the true $P(M)$; and 2) more importantly, the training and testing distributions may not be the same as previous methods have assumed, thus the weights derived from the training data would be essentially inappropriate for the test data. As illustrated in Figure 5.4, when training and test data have different distributions, $P(M)$ calculated using training data may be off from its true value, thus leading to the unsatisfactory performance of weighted ensemble (denoted as WE) as compared with the simple model averaging (AP). As follows, we formally illustrate why simple averaging with uniform weights beats other non-uniform weighting schemes.

Suppose the weights of k models are $\{w_1, w_2, \ldots, w_k\}$, each of which is from [0,1] and satisfies the constraint $\sum_{i=1}^{k} w_i = 1$. Ideally, the weight of model $M_i (1 \leq i \leq k)$ ought approximate its true probability $P(M_i)$ as well as possible. We use the following measure to evaluate the difference between the assigned weights and the true weights:

$$D(\mathbf{w}) = \sum_{i=1}^{k} (P(M_i) - w_i)^2 \tag{5.1}$$

Let Θ_i be the hypothesis space where M_i is drawn, which has a uniform distribution with a constant density C^i. In other words, we don't have any prior knowledge about the optimality of a model for a constantly changing stream. This is a valid assumption since the choice of optimal model is changing with the evolving distribution. But we are not assuming that we know anything about the future. The test distribution is somewhat revealed by the training distribution but which model fits the distribution the best remains unknown. Another clarification is that $P(M_i) \neq P(M_j) (i \neq j)$ on a specific pair of training and test sets given in time. This means that we cannot have preference for some model over others, since the preference needs to change continuously considering all possible training and test sets in time. The constraint $\sum_{i=1}^{k} P(M_i) = 1$ should also be satisfied. As an example, suppose there are two models, M_1 and M_2. Then $P(M_1)$ and $P(M_2)$ are both uniformly distributed within [0,1]. At one evaluation point, $P(M_1) = 0.8$ and $P(M_2) = 0.2$, but at another time, $P(M_1) = 0.3$ and $P(M_2) = 0.7$. It is clear that both M_1 and M_2 would be preferred at some time but it is unknown when and how this preference is changing. As another example, look at Figure 5.4 again, it is clearly shown that M_2 and M_1 are the best models with lowest test errors at time A and B respectively. However, since the labels of test examples are not known in advance, we could never know this changing preference before mining.

Integrating the distance measure in Eq. (5.1) over all possible M_i, we could obtain the expected distance as:

$$\begin{aligned} E[D(\mathbf{w})] =& \sum_{i=1}^{k} \int_{\Theta_i} C^i (P(M_i) - w_i)^2 \mathrm{d}M_i \\ =& \sum_{i=1}^{k} \int_{\Theta_i} C^i (P(M_i)^2 - 2P(M_i)w_i + w_i^2) \mathrm{d}M_i \end{aligned}$$

The aim is to minimize $E[D(\mathbf{w})]$ w.r.t $\mathbf{w}$. Eliminating irrelevant items, the above could be simplified to:

$$E[D(\mathbf{w})] = C_1 - C_2 \sum_{i=1}^{k} w_i + C_3 \sum_{i=1}^{k} w_i^2 \tag{5.2}$$

where $\{C_1, C_2, C_3\}$ are positive constants. Since $\sum_{i=1}^{k} w_i = 1$, the problem is transformed to:

$$\text{Minimize } \sum_{i=1}^{k} w_i^2 \text{ Subject to } \sum_{i=1}^{k} w_i = 1 \text{ and } 0 \le w_i \le 1$$

The closed form solution to this constrained optimization problem is: $w_i = \frac{1}{k} (1 \le i \le k)$. Therefore, when we have no prior knowledge about each model, equal weights are expected to be the closest to true model probabilities on the test data over some period of time, thus giving the best performance on average. This is particularly true in the stream environment where the distribution is continuously changing. As shown in the following experiments, the best model on current data may have bad performance on future data, in other words, $P(M)$ is changing and we could never estimate the true $P(M)$ and when and how it would change. Hence non-uniform weights could easily incur overfitting, and relying on a particular model should be avoided. Under such circumstances, uniform weights for the models are the best approximate of the true $P(M)$.

5.4 Classifying Imbalanced Data Streams

In this section, we propose a simple strategy that can effectively mine data streams with skewed distribution. The choice of methods incorporates the analysis made in section 5.3.

Stream Ensemble Framework Skewed distribution can be seen in many data stream applications. In these cases, the positive examples are much less popular than the negative ones. Also, misclassifying a positive example usually invokes a much higher loss compared to that of misclassifying a negative example. Therefore, the traditional inductive learner, which tends to ignore positive examples and predict every example as negative, is undesirable for skewed stream mining. We propose a simple, systematic method to handle skewed data streams. We will start with problem definition, and then present the algorithm.

In some applications such as credit card application flow, the incoming data stream arrives in sequential chunks, $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of the same size n. $\mathbf{S}_m$ is the most up-to-date chunk. The data chunk that arrives next is $\mathbf{S}_{m+1}$, and for simplicity, we denote it as $\mathbf{T}$. The aim of stream classification is to train a classifier based on the data arrived so far to estimate posterior probabilities of examples in $\mathbf{T}$. We further assume that the data comes from two classes, positive and negative classes, and the number of examples in negative class is much greater than the number of positive examples. In other words, $\mathcal{P}(+) \ll \mathcal{P}(-)$. In this two-class problem, only the posterior probability of positive class $\mathcal{P}(+|\mathbf{x})$ is computed, then that of the negative class is simply $1 - \mathcal{P}(+|\mathbf{x})$. To have accurate probability estimation, we propose to utilize both sampling and ensemble techniques in our framework.

Sampling. We split each chunk $\mathbf{S}$ into two parts $\mathbf{P}$, which contains positive examples in $\mathbf{S}$, and $\mathbf{Q}$, which contains negative examples in $\mathbf{S}$. The size of $\mathbf{P}$ is much smaller than that of $\mathbf{Q}$. For example, in network intrusion detection data, there are 60262 normal examples, but only 168 U2R attacks. Also, it should be noted that in stochastic problems, a given $\mathbf{x}$ could appear in both $\mathbf{P}$ and $\mathbf{Q}$ for several times. The count of $\mathbf{x}$ in each class will contribute to the calculation of posterior probability $\mathcal{P}(y|\mathbf{x})$.

In stream mining, we cannot use all data chunks as training data. First, stream data is huge in amount and it is usually impossible to store all of them. Second, stream mining requires fast processing, but a huge training set will make the classification process extremely slow, thus is unsatisfactory. Model reconstruction on new data usually reduces the expected error. In other words, the best way to construct a model is to build it upon the most recent data chunk. This works for examples in negative class since these examples dominate the data chunk and are sufficient for training an accurate model. However, the positive examples are far from sufficient. An inductive learner built on one chunk will perform poorly on positive class. To enhance the set of positive examples, we propose to collect all positive examples and keep them in the training set. Specifically, the positive examples in the training set is $\{\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_m\}$. On the other hand, we randomly under sample the negative examples in the last data chunk $\mathbf{Q}_m$ to make the class distribution balanced.

Ensemble. Instead of training a single model on this training set, we propose to generate multiple samples from the training set and compute multiple models from these samples. The advantage of ensemble is that the accuracy of multiple model is usually higher than that of a single model trained from the entire dataset. As shown in Section 5.3, the expected error could be reduced by training multiple models. The samples should be as uncorrelated as possible so that the base classifiers would make uncorrelated errors which could be eliminated by averaging. To get uncorrelated samples, each negative example in the training set is randomly propagated to exactly one sample, hence the negative examples in the samples are completely disjoint. As for positive examples, they are propagated to each sample. We take a parameter r as input, which is the ratio of positive examples over negative examples in each sample. r is typically between 0.3 to 0.6 to make the distribution balanced. Let n_p be the number of positive examples in the training set, then the number of negative examples in each sample is: $n_q = \lceil n_p/r \rceil$. Suppose k samples are generated, then a series of classifiers $C_1, C_2, \dots, C_k$ are trained on the samples. Each classifier C_i outputs an estimated posterior probability $f^i(\mathbf{x})$ for each example $\mathbf{x}$ in $\mathbf{T}$. We use simple averaging to combine probability outputs from k models:

$$f^E(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} f^i(\mathbf{x}) \tag{5.3}$$

It is worth noting that this differs from bagging: 1) in bagging, bootstrap samples are used as training sets, and 2) bagging uses simple voting while our framework generates averaged probability for each test example. The outline of the algorithm is given in Algorithm 3. We assume that each data chunk can fit the main memory.

Algorithm 3 Ensemble Algorithm for Skewed Stream Classification

```
Input: Current data chunk $\mathbf{S}$, test data $\mathbf{T}$, number of ensembles $k$, distribution ratio $r$, set of positive examples $\mathbf{AP}$
Output: Updated set of positive examples $\mathbf{AP}$, posterior probability estimates for examples in $\mathbf{T}$, $\{f^E(\mathbf{x})\}_{\mathbf{x}\in\mathbf{T}}$.
Algorithm:
  1. Split $\mathbf{S}$ into $\mathbf{P}$ and $\mathbf{Q}$ according to their definitions.
  2. Update $\mathbf{AP}$ as $\{\mathbf{AP}, \mathbf{P}\}$
  3. Calculate the number of negative examples in the sample $n_q$ based on the values of $r$ and $n_p$.
  4. for $i = 1$ to $k$ do
      (a) Draw a sample of size $n_q$ from $\mathbf{Q}$ without replacement, $\mathbf{O}$.
      (b) Train a classifier $C_i$ on $\{\mathbf{O}, \mathbf{AP}\}$.
      (c) Compute posterior probability estimates $\{f^i(\mathbf{x})\}_{\mathbf{x}\in\mathbf{T}}$ using $C_i$
  5. Compute posterior probability estimates by combining ensemble outputs $\{f^E(\mathbf{x})\}_{\mathbf{x}\in\mathbf{T}}$ based on Eq. (5.3).
```

Error Reduction by Sampling We explain how the use of sampling techniques contributes to error reduction. First, we give some background in error decomposition. We expect that a well trained classifier could approximate the posterior class distribution. However, the estimate of posterior probability is not necessarily the true probability. Therefore, in classification, besides Bayes error, there are remaining errors, which could be decomposed into bias and variance. The bias measures the difference between the expected probability and the true probability, whereas the variance measures the changes in estimated probabilities using varied training sets. As stated in [159, 162], given $\mathbf{x}$, the output of a classifier can be expressed as:

$$f_c(\mathbf{x}) = \mathcal{P}(c|\mathbf{x}) + \beta_c + \eta_c(\mathbf{x}) \tag{5.4}$$

where $\mathcal{P}(c|\mathbf{x})$ is the posterior probability of class c given input $\mathbf{x}$, β_c is the bias introduced by the classifier and $\eta_c(\mathbf{x})$ is the variance of the classifier given input $\mathbf{x}$.

In two-class problem, $\mathbf{x}$ is assigned to positive class if $\mathcal{P}(+|\mathbf{x}) > \mathcal{P}(-|\mathbf{x})$. The Bayes optimal boundary is therefore represented by a set of points $\mathbf{x}^*$ that satisfy $\mathcal{P}(+|\mathbf{x}^*) = \mathcal{P}(-|\mathbf{x}^*)$. However, since $f_c(\mathbf{x})$ is different from $\mathcal{P}(c|\mathbf{x})$, the estimate of Bayes boundary is incorrect, the boundary error is $b = \mathbf{x}_b - \mathbf{x}^*$ where $\mathbf{x}_b$ are the estimated boundary points that have $f_+(\mathbf{x}_b) = f_-(\mathbf{x}_b)$. In [159], it shows that classification error rate is linearly proportional to the boundary error. So we will focus on the analysis of boundary error from now on. In analogy with bias-variance decomposition described in Eq. (5.4), the boundary error can be expressed in term of boundary bias and boundary variance:

$$b = \frac{\eta_+(\mathbf{x}_b) - \eta_-(\mathbf{x}_b)}{s} + \beta_b \tag{5.5}$$

where $s = p'_+(\mathbf{x}^*) - p'_-(\mathbf{x}^*)$ is independent of the trained model, and β_b is $(\beta_+ - \beta_-)/s$. If $\eta_c(\mathbf{x})$ is independent and has Gaussian distribution with zero mean and variance $\sigma^2_{\eta_c}$. Then b is also normally distributed with mean β_b and variance σ^2_b where

$$\sigma^2_b = (\sigma^2_{\eta_+} + \sigma^2_{\eta_-})/s^2 \tag{5.6}$$

Now we show that sampling techniques in the proposed framework reduces variance in skewed data classification. Our sampling approach could reduce σ^2_b not at the expense of increase in β_b. If only the current data chunk is used to train the model, the positive examples are so limited that the error of the classifier would mainly come from the variance. In the proposed framework, the positive examples in the previous time shots are incorporated into the training set. Adding positive examples would reduce the high variance σ^2_b caused by insufficient data. When there are concept changes, the bias may be affected by adding old examples, but it may increase very slightly. The reason is that the negative examples of the training set are from the current data chunk, which are assumed sufficient and reflecting the current concept. Therefore, the boundary between the two classes could not be biased much by including old positive examples in the training set. Even if the bias β_b is increasing, the reduction of variance is dominant and the overall generalization accuracy is improved.

5.5 Experiments

We conduct an extensive performance study using both synthetic and real data sets, where training and testing distributions are explicitly generated differently, to demonstrate the effectiveness of the averaging ensemble against change. As discussed below in detail, this empirical study validates the following claims: 1) ensemble based on model averaging would reduce expected errors compared with single models, thus is more accurate and stable; and 2) previous weighted ensemble approach is less effective than ensemble based on simple voting or probability averaging.

5.5.1 Experiment Setup

Synthetic Data Generation We describe how to generate synthetic data and simulate its concept changes as follows.

Form of $P(\mathbf{x})$. $\mathbf{x}$ follows a Gaussian distribution, i.e., $P(\mathbf{x}) \sim N(\mu, \mathbf{\Sigma})$, where μ is the mean vector and $\mathbf{\Sigma}$ is the covariance matrix. The feature change is simulated through the change of the mean vector where μ_i is changed to $\mu_i s_i(1+t)$ for each data chunk. t is between 0 to 1, representing the magnitude of changes, and $s_i \in \{-1, 1\}$ specifies the direction of changes and could be reversed with a probability of 10%.

Form of $P(y|\mathbf{x})$ in deterministic problems. In binary problems, the boundary between two classes is defined using function $g(\mathbf{x}) = \sum_{i=1}^{d} a_i x_i x_{d-i+1} - a_0$ where $\mathbf{a}$ is the weight vector. Then the examples satisfying $g(\mathbf{x}) < 0$ are labeled positive, whereas other examples are labeled negative. a_i is initialized by a random value in the range of [0,1]. The value of a_0 is set based on the values of $\{a_1, \ldots, a_d\}$ and controls the degree of skewness. In multi-class problems, suppose there are l classes and the count of examples in each class is $\{C_1, C_2, \ldots, C_l\}$. We calculate the value of $g(\mathbf{x})$ for each $\mathbf{x}$ using the definition given in binary problems. All the examples are ranked in ascending order of $g(\mathbf{x})$. Then the top C_1 examples are given class label 1, examples with ranks $C_1 + 1$ to $C_1 + C_2$ are assigned to class 2, and so on. In both problems, the concept change is represented by the change in weight a_i, which is changed to $a_i s_i(1 + t)$ for every data chunk. The parameters t and s_i are defined in the same way as in the feature change.

Form of $P(y|\mathbf{x})$ in stochastic problems. We use a sigmoid function to model the posterior distribution of positive class: $P(+|\mathbf{x}) = 1/(1 + \exp(g(\mathbf{x})))$. The concept changes are also realized by the changes of weights as illustrated in the deterministic scenario. The skewness in binary classification problems is also controlled by a_0 in $g(\mathbf{x})$.

Real-World Data Sets We test several models on KDD Cup'99 intrusion detection data set, which forms a real data stream. This data set consists of a series of TCP connection records, each of which can either correspond to a normal connection or an intrusion. We construct two data streams from the 10% subset of this data set:

Shuffling. Randomly shuffle the data and partition it into 50 chunks with varying chunk size from 5000 to 10000.

Stratified Sampling. Put the data into class buckets: One for normal connections and one for intrusions. Generate 50 chunks as follows: 1) choose an initial $P(y)$, 2) sample without replacement from each bucket to form a chunk that satisfies $P(y)$, 3) evolve $P(y)$ and sample from the remaining data in the buckets as the next chunk, and finally, 4) put data sampled in step 2 back to the buckets and repeat steps 2 and 3. The chunk size is also varied from 5000 to 10000.

Measures and Baseline Methods For a data stream with chunks $T_1, T_2, \ldots, T_N$, we use T_i as the training set to classify T_{i+1} and the distribution of the test set T_{i+1} is not necessarily the same as that of T_i. We evaluate the accuracy of each model. For the classifier having posterior probability as the output, the predicted class label is the class with the highest posterior probability under zero-one loss function. Another measure is mean squared error (MSE), defined as the averaged distance between estimated probability and true posterior probability $P(y|\mathbf{x})$. In problems where we are only exposed to the class labels but do not know the true probability, we set $P(y|\mathbf{x}) = 1$ if y is $\mathbf{x}$'s true class label, otherwise $P(y|\mathbf{x}) = 0$. We are comparing the following algorithms: single models built using Decision Tree (DT), SVM, Logistic Regression (LR) and ensemble approaches including Weighted Ensemble (WE), Simple Voting (SV) and Averaging Probability (AP). Different from averaging ensemble framework, the weighted ensemble approach assigns a weight to each base model which reflects its predictive accuracy on the training data (obtained by ten-fold cross validation) and the final prediction outputs are combined through weighted averaging. In previous work, such weighted ensembles are shown to be effective when the "shared distribution" assumption holds true. In our experiments, we evaluate its performance upon the relaxed assumption. For all the base learning algorithms, we use the implementation in Weka package [172] with parameters set to be the default values. In the averaging ensemble framework, either SV or AP, the base streaming models could be chosen arbitrarily. We test the framework where base models are constructed from either different learning algorithms or different samples of the training sets.

For a learning algorithm A_h, we build a model based on T_i and evaluate it on T_{i+1} to obtain its accuracy p_{ih} and MSE e_{ih}. There are altogether $N - 1$ models and we report its average accuracy (Aacc) and average MSE (Amse). Furthermore, in each of the $N - 1$ runs, we compare the performance of all algorithms and decide the winner and loser in the following way: if p_{ih} is within $m\%$ of $\max_h p_{ih}$, algorithm A_h is a winner in that run, similarly, if p_{ih} is within $m\%$ of $\min_h p_{ih}$, it is a loser. In other words, we tolerate some small difference between two algorithms, if their accuracies are the same with respect to the "margin tolerance rate" m, we regard their performance as the same. We report the number of wins and loses for each algorithm (#W and #L). With winners ranking the first, losers ranking the third and all other algorithms occupying the second position, we give $N-1$ ranks to each algorithm and obtain the mean and standard deviation of the ranks (AR and SR). A good algorithm will have a higher accuracy, a lower MSE and average rank closer to 1. If it has a lower standard deviation in the ranks, the learning algorithm is more stable.

5.5.2 Empirical Results

We report the experimental results comparing the two ensemble approaches (SV, AP) with single model algorithms (DT, SVM, LR) as well as weighted ensemble method (WE). As discussed below in detail, the results clearly demonstrate that on the stream data where training and testing distributions are different and fast evolving, the two ensemble approaches have the best performance on average with higher accuracy and lower variations. Therefore, when facing unknown future, the ensemble framework is the best choice to minimize the number of bad predictions.

Test on Concept-Drifting Stream Data We generate four synthetic data streams, each of which is either binary or multi-class and has chunk size 100 or 2000. The distribution within a data chunk is unchanged whereas between data chunks, the following changes may occur: 1) each data chunk could either be deterministic or stochastic (in binary problem); 2) in each chunk, the Gaussian distribution of the feature values may either have diagonal variance matrix or non-diagonal one; 3) either one of the three concept changes (feature change, conditional change and dual change) may occur; 4) the number of dimensions involved in the concept change is a random number from 2 to 6; and 5) the magnitude of change in each dimension is randomly sampled from $\{10\%, 20\%, \ldots, 50\%\}$. Since lots of random factors are incorporated into the simulated concept change, it is guaranteed that training and testing distributions are different and evolving quickly. Each data set has 10 dimensions and 100 data chunks. The margin tolerance rate is set to be 0.01.

From Table 5.1, it is clear that the two ensemble approaches (SV and AP) have better perfor-

Table 5.1: Performance Comparison on Synthetic Stream Data

Binary Stream Data

Measure	Chunk Size 100						Chunk Size 2000					
	DT	SVM	LR	WE	SV	AP	DT	SVM	LR	WE	SV	AP
Aacc	0.7243	0.7591	0.7346	0.7461	0.7595	**0.7690**	**0.8424**	0.8318	0.8366	0.8339	0.8370	0.8369
Amse	0.2731	0.2387	0.2625	0.1889	0.2379	**0.1752**	0.1540	0.1649	0.1601	0.1262	0.1597	**0.1242**
AR	2.2323	1.6465	2.1111	1.8889	1.5152	**1.4848**	2.1313	1.8485	1.6869	1.7980	**1.5455**	**1.5455**
SR	0.8902	0.6898	0.8193	0.7544	**0.5414**	**0.5414**	0.9757	0.8732	0.8765	0.8687	**0.7460**	**0.7460**
#W	30	47	28	34	50	**53**	41	46	58	49	**60**	**60**
#L	53	12	39	23	**2**	**2**	54	31	27	29	**15**	**15**

Multi-Class Stream Data

Measure	Chunk Size 100						Chunk Size 2000					
	DT	SVM	LR	WE	SV	AP	DT	SVM	LR	WE	SV	AP
Aacc	0.5111	0.5295	0.5298	0.5301	**0.5320**	0.5314	0.4991	0.4939	0.4920	0.5130	0.4950	**0.5139**
Amse	0.1745	0.1413	0.1272	0.1210	0.1872	**0.1208**	0.1764	0.1461	0.1322	0.1246	0.2020	**0.1244**
AR	2.3636	1.9293	1.9798	1.8283	1.8788	**1.7273**	2.0202	2.2626	2.2424	1.6667	2.1111	**1.4040**
SR	0.8263	0.7986	0.7822	**0.5159**	0.6589	0.6197	0.8919	0.8758	0.9045	**0.4949**	0.9023	0.5330
#W	22	35	31	23	28	**36**	38	28	31	34	35	**61**
#L	58	28	29	**6**	16	9	40	54	55	**1**	46	2

mance (best are highlighted in bold font) regardless of the measures we are using, the problem type (binary or multi-class) and the chunk size. Take the binary problem with chunk size 100 as an example. AP proves to be the most accurate and stable classifier with the highest accuracy (0.7690), lowest MSE (0.1752), 53 wins and only 2 loses. SV is quite comparable to AP with 50 wins and 2 loses. The best single classifier SVM wins 47 times and loses 12 times and WE approach seems to suffer from its training set-based weights with only 34 wins but 23 loses. These results suggest the following: when the "same distribution" between training and testing data does not exist: 1) there are no uniformly best single classifiers, even decision tree, which has the worst average performance, still wins 30 times among all 99 competitions. The large variabilities of single models result in their high expected errors; 2) on average, ensemble approaches, simple voting or probability averaging, are the most capable of predicting on future data with unknown distributions; 3) assigning a weight to each base learner even hurts the predictive performance on testing data since the distribution it tries to match is different from the true one.

For binary streams, we also record the results on the first 40 chunks to see how the concept evolution affects the classification performance. The results indicate that even within the same data stream, the best single classifier for the first 40 chunks is different from the best one on the whole data set. Take the stream data with chunk size 100 as an example. At first, LR has 18 wins, compared with DT (4 wins) and SVM (14 wins), it appears to be the best on average. However, later, SVM takes the first place with 47 wins (DT 30 and LR 28). This clearly indicates that in a stream whose distribution evolves, a model which performs well on current data may have poor performance on future data. Since we never know when and how the distribution changes,

Table 5.2: Ensemble on Real Data

Shuffling

	DT	SVM	LR	WE	SV	AP
Aacc	0.9961	0.9941	0.9957	0.9964	**0.9975**	**0.9975**
Amse	0.0039	0.0059	0.0043	0.0028	0.0025	**0.0024**
AR	1.9592	2.5306	1.9388	1.6939	**1.0000**	**1.0000**
SR	0.8406	0.7665	0.8013	0.7959	**0**	**0**
#W	18	8	17	25	**49**	**49**
#L	16	34	14	10	**0**	**0**

Stratified Sampling

	DT	SVM	LR	WE	SV	AP
Aacc	0.9720	0.9744	0.9699	0.9707	**0.9755**	**0.9755**
Amse	0.0280	0.0256	0.0301	0.0259	0.0245	**0.0232**
AR	1.6531	1.5510	1.6122	1.5306	**1.2245**	**1.2245**
SR	0.9026	0.7654	0.8854	0.8191	**0.4684**	**0.4684**
#W	31	30	32	33	**39**	**39**
#L	14	8	13	10	**1**	**1**

depending on one single classifier is rather risky. On the other hand, ensemble based on averaged probability is more robust and accurate, which is the winner for classifying data streams with regard to the average performance (ranks around 1.5 while others rank more than 2 on average). Ensemble based on simple voting (SV) produces results similar to that of AP in binary stream problems, but is not that competitive in multi-class problems. The reason may be that two class problems are easier for prediction tasks, so the probability outputs of a classifier may be rather skewed, greater than 0.9 or less than 0.1. So there isn't much difference between simple voting and averaging probability in this case. However, when the number of classes grows large, it is quite unlikely that the predicted probability is skewed. The strengths of probability averaging over simple voting is therefore demonstrated on multi-class problems. As for the weighted ensemble approach, it sometimes increases the predictive accuracy, but sometimes gives even worse predictions compared with single models. Whether it performs good or not is dependent on how the training and testing distributions match. In this sense, the other two simple ensemble methods are more robust since they are not based on the assumption that training and testing data come from the same distribution.

Test on KDD Cup'99 Data In Section 5.5.1, we describe the two data streams we generate from the KDD Cup'99 intrusion detection data set and how the training and testing distributions

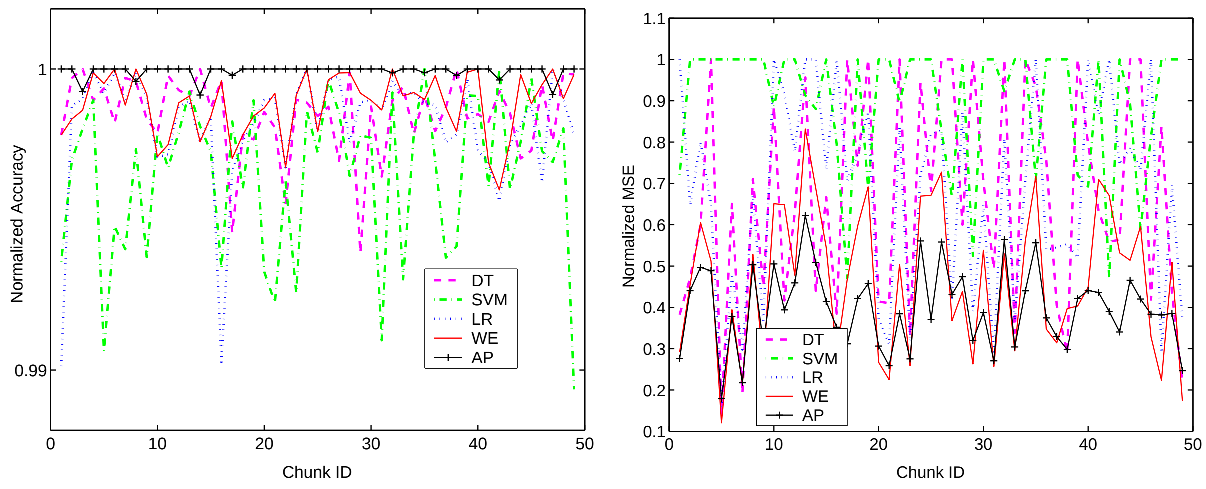


Figure 5.5: Accuracy and Mean Squared Error on Real Data

are made different explicitly. Also, as illustrated in Section 5.2, both $P(\mathbf{x})$ and $P(y|\mathbf{x})$ undergo continuous and significant changes in this stream data. Results of various methods on the two streams are summarized in Table 5.2 where margin tolerance rate is set to be 0.001. Similar to earlier results on simulated streams, the advantage of the ensemble framework is clearly demonstrated. The two ensemble approaches not only increase the accuracy of single models but also occupy the first place in most of the evaluations. The most significant improvements could be observed on the data set generated by shuffling, where accuracy goes up from 0.9961 to 0.9975 and the number of wins increases from 18 to 49 after combining outputs of multiple models. The performance of SV and AP is almost the same for these two data sets. As discussed in the synthetic data experiments, SV and AP are expected to have similar predictions when the estimated probabilities of each class are skewed in binary problems. Another observation is that the weighted ensemble approach could improve over a single model but the improvements are less significant compared with simple averaging. This phenomenon again shows that the weighting scheme cannot survive the relaxed assumption where training and testing distributions could be different since it fits the training data too “tightly”.

Figure 5.5 reveals some detailed information about the evaluation results (Accuracy and MSE w.r.t Chunk ID) on the first data set where data records are randomly shuffled. To exclude the effects of different scales, we normalize the measures by the maximal value. It is obvious that the probability averaging ensemble (AP) is the most accurate classifier in general with normalized accuracy close to 1 and mean squared error below 0.5. Also, as shown in both plots, as measures of single models fluctuate within a wide range, the performance of probability averaging ensemble is much more stable. This clearly shows the benefits of using our ensemble framework when the testing distribution is unknown and departed from the training distribution. On average, the ensemble would approximate the true distribution more accurately than single models, with least number of loses. The weighted ensemble could achieve higher accuracy than single-model classifier but still has larger variance and worse average performance compared with AP. For example, the highest normalized MSE of AP is only around 0.6, but over 0.8 for weighted ensemble approach.

Test on Skewed Data Streams

We focus on binary classification problems with skewed distributions and generate synthetic data streams with different kinds of concept drifts. Six kinds of stream data sets are generated, each of which is either deterministic or stochastic, and has either feature, conditional, or dual concept changes. Each data set has 10 dimensions, 11 chunks with chunk size 1000. The percentage of rare examples is 1%, and for each data chunk, two dimensions are chosen randomly to change 10%. The last chunk in the data set is recognized as the test data, and all other data chunks are used for training. Since we are more interested in probability estimates of the positive class, the mean square error of the positive class is reported for each kind of stream data. The results are obtained by calculating errors of 10 randomly generated data sets. We use C4.5, Naive Bayes, and logistic regression as base learners. We would evaluate the proposed method that handle skewed streams in two perspectives: (1) are the probability estimates accurate? and (2) is the classification accurate?

Evaluation Criteria There are some standard measures for evaluating the quality of probability estimation. A popular one is mean squared error, defined as:

$$L = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - \mathcal{P}(+|\mathbf{x}_i))^2 \tag{5.7}$$

where $f(\mathbf{x}_i)$ is the output of ensemble, which is the estimated posterior probability of $\mathbf{x}_i$, and $\mathcal{P}(+|\mathbf{x}_i)$ is the true posterior probability of $\mathbf{x}_i$. Since in skewed mining problems, rare class is more interesting and usually associated with higher classification cost, we would like to have a low L for examples in the rare class.

In skewed mining problems, classification error is not a good measure since the examples in the majority class will dominate the result, and it is hard to tell whether rare examples are classified correctly. Therefore, for this kind of problems, the following evaluation metrics are typically used: Precision (Detection Rate), Recall, and False Alarm Rate. To show how these metrics are correlated, we use both ROC curve and recall-precision plot to demonstrate the experimental results. The ROC curve represents the trade-off between detection rate and false alarm rate and plots a 2-D graph, with x-axis as the false alarm rate and y-axis as the detection rate. The ideal ROC curve has 0% false alarm rate and 100% detection rate. In other words, the area under ROC curve is 1 in the ideal case. Therefore, a good algorithm would produce a ROC curve as close to the left-top corner as possible. So the area under ROC curve (AUC) is an evaluation metric, where a better algorithm will have an AUC value closer to 1. Another method to evaluate the results is to plot the correlation between recall and precision. The recall-precision plot will have precision as the y axis and recall as the x axis.

Baseline Methods We show that both sampling and ensemble techniques could help reduce the classification error. Therefore, the baseline methods we are comparing with are:

No Sampling + Single Model (*NS*). Only the current data chunk is used for training, which is highly skewed. A single model is trained on the training set.

Sampling + Single Model (*SS*). The training set is the same as that used in our proposed ensemble methods. It is obtained by keeping all positive examples seen so far and under sampling negative examples in the current data chunk. The difference lies in the classification model which is a single model in this method, but a multiple model in our proposed method.

Accordingly, we denote our method as *Sampling + Ensemble* (*SE*), which adopts both sampling and ensemble techniques. By comparing with the above two baseline methods, the strengths of sampling and ensemble could be illustrated well.

In the experiments, the base learners include both parametric and non-parametric classifiers: Decision Tree, Naive Bayes and Logistic Regression. We use the implementation in Weka package [172]. The parameters for single and ensemble models are set to be the same, which are the default values in Weka.

Table 5.3: Mean Squared Error on Deterministic Stream Data

Changes	Decision Trees			Naive Bayes			Logistic Regression		
	SE	NS	SS	SE	NS	SS	SE	NS	SS
Feature	0.1275	0.9637	0.6446	**0.0577**	0.8693	0.4328	0.1501	0.8117	0.5411
Conditional	0.0943	0.9805	0.5500	**0.0476**	0.8830	0.4380	0.1301	0.8944	0.5729
Dual	0.0854	0.9521	0.5174	**0.0664**	0.8596	0.4650	0.1413	0.8371	0.5525

Table 5.4: Mean Squared Error on Stochastic Stream Data

Changes	Decision Trees			Naive Bayes			Logistic Regression		
	SE	NS	SS	SE	NS	SS	SE	NS	SS
Feature	0.0847	0.6823	0.4639	**0.0314**	0.5371	0.2236	0.0974	0.5311	0.3217
Conditional	0.0552	0.6421	0.4463	**0.0299**	0.5675	0.2449	0.1029	0.6578	0.4151
Dual	0.0684	0.6758	0.4107	**0.0301**	0.5981	0.2556	0.0887	0.6388	0.4075

Empirical Results In this part, we report the experimental results regarding the effectiveness and efficiency of our proposed method. The results are shown in Table 5.3 (deterministic) and Table 5.4 (stochastic), respectively.

It is clearly seen that, no matter how the concept changes, our proposed method (SE) greatly improves the mean square error of the positive class in both deterministic and stochastic data streams. The decrease in error rate is significant, from 0.9 to 0.1 on the deterministic data, and from 0.6 to 0.06 on the stochastic data on average. NS performs badly since only the current data chunk is used for training and it is highly skewed. When training on a skewed data set, the inductive learner would build a model that tends to ignore positive examples and simply classify every example as negative. Therefore, NS generates an error close to 1 regarding mean square error on the positive class. According to the performance of SS, the mean square error of the positive class is reduced to around 0.5 after we oversample positive examples and under sample negative examples. The reason is that the class distribution is more balanced after incorporation of more positive examples. This helps improve the performance on the positive class. However, the single model would still have a high error rate caused by classification variance.

Although SE utilizes exactly the same training sets as used in SS, the performance of SE is much better since the ensemble could reduce the variance of classifier by averaging the outputs. As seen in Tables 5.3 and 5.4, for our proposed method, the mean square error on the positive class is usually less than 0.1. The most significant reduction in error rate is 0.45 (SE vs. SS) and 0.88 (SE vs. NS). On average, the error decreases around 40% after using sampling and further reduces to 10%-20% if we use both sampling and ensemble.

It can be observed that Naive Bayes has the best performance on synthetic data set. This is due to the fact that synthetic data is generated using a Gaussian distribution with a diagonal covariance matrix, which guarantees independence between features.

Thus we conclude that our proposed method consistently improves posterior probability estimates of minority class under feature, conditional, and dual concept drifts in both deterministic and stochastic applications.

5.6 Related Work

Sample selection bias [178] investigates the effect on learning accuracy when the training data is a "biased" sample of the true distribution. Although the true target function to be modeled, $P(y|\mathbf{x})$, does not "explicitly" change, its value can be wrong in various ways in the biased training data. Previously, decision tree based model averaging has been shown to be helpful to correct feature bias or the bias where the chance to sample an example into the training set is independent on y given $\mathbf{x}$ [178]. The most important difference of our work from these previous studies is: (1) $P(y|\mathbf{x})$ in our problem is allowed to explicitly change and can change significantly, (2) changes in $P(y|\mathbf{x})$ are combined with changes in $P(\mathbf{x})$. To consider the significance of our work under sample selection bias formulation, our comprehensive results significantly extend the previous work and demonstrate that model averaging can reliably correct sample selection bias where biased conditional probability is quite different from unbiased testing data.

Class imbalance has become an important research problem in recent years since more people have realized that imbalance in class distribution causes suboptimal classification performance [169, 34]. Many solutions have been proposed to handle this problem by preprocessing data, transforming algorithms or post-processing models [1]. Among them, balancing training set distribution is the most popular approach. Specifically, many sampling algorithms have been developed to either under sample majority examples or oversample minority examples [11, 1]. These methods may improve the prediction accuracy of minority class, however, they are greatly challenged by stream applications where infinite data flow and continuous concept drifts are present. Therefore, a general framework for dealing with skewed data stream is in great demand.

Skewed stream problems have been studied in the context of summarization and modeling [38, 107]. However, the evaluation of existing stream classification methods is done on balanced data streams [88, 162, 49, 2]. In reality, the concepts of data streams usually evolve with time. Several stream classification models are designed to mine such concept-drifting data streams [162, 49], however, they regard concept drifts as changes in conditional probability. In our work, it is shown that concept changes may occur in both feature and conditional probability. In [52, 180], two application examples of skewed data mining are studied. But we provide a more general framework for building accurate classification models on skewed data streams.

5.7 Summary

In this chapter, we demonstrate that the distributions of stream data can evolve in some unknown manner, and models matching training distribution well may perform poorly in continuously changing distributions. In order to design robust and effective stream mining algorithms against changes, an appropriate methodology is not to overly match the training distribution, such as by weighted voting or weighed averaging where the weights are assigned according to training distribution. On these basis, we propose an ensemble framework based on model averaging of conditional probability estimators. We demonstrate both formally and empirically such a framework can reduce expected errors and give the best performance on average when the test data does not follow the same distribution as the training data. Since the property of expected error reduction is proven formally, the framework is expected to have robust and better performance regardless of chosen baseline models. In particular, we extend the framework using sampling techniques to mine skewed data streams. The algorithm first generates a balanced training set by keeping all positive examples and under sampling negative examples. Then the training set is further divided into several samples and multiple models are trained on these samples. The final outputs are the averaged probability estimates on test data by multiple models. The error reduction is significant according to experimental results.

Part II

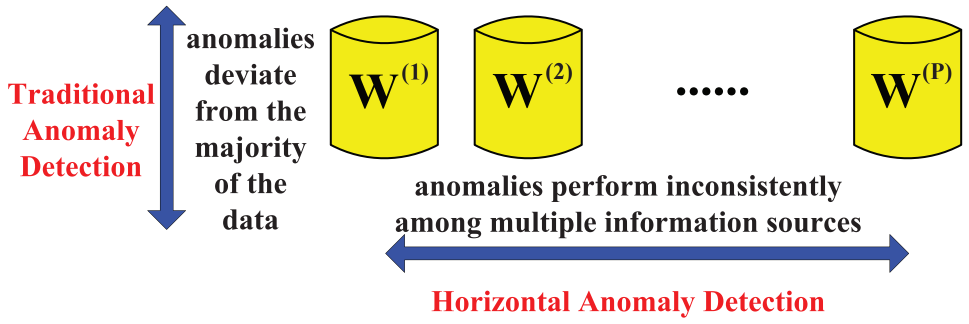
Inconsistency Detection

Chapter 6

Inconsistency Detection across Multiple Heterogeneous Sources

In Part I, we presented our contributions to the field of consensus combination for effective multi-source classification. Reaching consensus among heterogeneous information sources gives us the gains in classification performance. On the other hand, by exploring the *differences* among sources, we can identify something unusual and interesting, and thus the second part of this thesis is to detect anomalies or inconsistencies across multiple information sources. Although the problem of anomaly detection has been widely studied [31], most of the existing approaches identify anomalies from one single data source. Different from existing work, we propose to identify inconsistencies across multiple information sources as a new type of meaningful anomalies in this part. In this chapter, we define the general problem of inconsistency detection across multiple heterogeneous sources and propose an effective spectral framework to identify such anomalies. In Chapters 7 and 8, we propose effective inconsistency detection solutions for information network analysis and system debugging problems.

In this chapter, we propose to detect objects that have inconsistent behavior among multiple heterogeneous sources [64]. A set of objects can be described from various perspectives (multiple information sources). The underlying clustering structure of normal objects is usually shared by multiple sources. However, anomalous objects belong to different clusters when considering different aspects. To identify such objects, we compute the distance between different eigen decomposition results of the same object with respect to different sources as its anomalous score, and give interpretations from the perspectives of constrained spectral clustering and random walks over graph. Experimental results on several UCI as well as DBLP and MovieLens datasets demonstrate the effectiveness of the proposed approach. The proposed method can detect anomalies that cannot be found by traditional anomaly detection techniques and provide new insights into the application area.



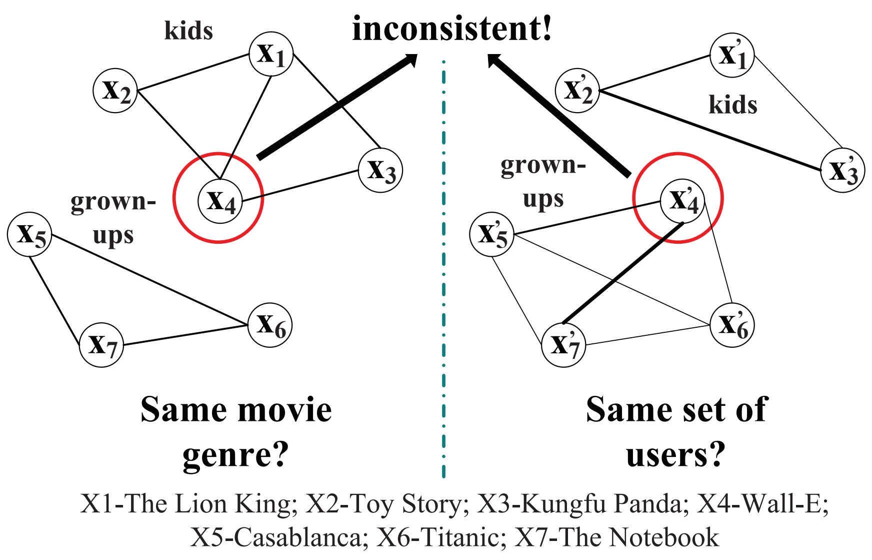
Traditional Anomaly Detection
anomalies deviate from the majority of the data
$W^{(1)}$
$W^{(2)}$
......
$W^{(P)}$
anomalies perform inconsistently among multiple information sources
Horizontal Anomaly Detection

Figure 6.1: Illustration of Horizontal Anomaly Detection Problem

6.1 Overview

Today's information explosion generates significant challenges for anomaly detection when there exist many large, distributed data repositories consisting of a variety of data sources and formats. While traditional anomaly detection approaches focus on identifying objects that are dissimilar to most of the other objects from a single source [31], we aim at detecting objects that have "inconsistent behavior" among multiple information sources, which we refer to as "horizontal anomaly detection". Distinction between traditional anomaly detection and horizontal anomaly detection is shown in Figure 6.1 where traditional anomaly detection explores the single information source vertically and horizontal anomaly detection explores horizontally the inconsistencies among multiple information sources instead. In the following discussions, we will give a few practical examples.

In today's information age, there are usually several sources of information that describe different properties or characteristics of individual objects. For example, we can learn about a movie from its basic information including genre, cast, plots, etc., or the tags users give to the movie, or the viewing histories of the users who watched the movie. On each of the information source, a relationship graph can be derived to characterize the pairwise similarities between objects where the edge weight indicates the degree of similarity. As an example, Figure 6.2 shows the similarity relationships among a set of movies derived from two information sources: movie genres and users. The genre information may indicate that two movies that are "animations" are more similar than two movies one of which is an "animation" and one of which is a "romance" movie. Similarly, movies watched by the same set of users are likely to be more similar than movies that are watched by completely different sets of users.



inconsistent!
kids
grown-ups
kids
grown-ups
x_1 x_2 x_3 x_4 x_5 x_6 x_7
x'_1 x'_2 x'_3 x'_4 x'_5 x'_6 x'_7
Same movie genre?
Same set of users?
X1-The Lion King; X2-Toy Story; X3-Kungfu Panda; X4-Wall-E;
X5-Casablanca; X6-Titanic; X7-The Notebook

Figure 6.2: A Horizontal Anomaly Detection Example based on Movie Similarity

Clearly, objects form a variety of clusters or communities based on each similarity relationship. For example, two clusters can be found from both of the similarity graphs in Figure 6.2. One cluster represents the movies that are animations, which are loved by kids; while the other cluster represents romance movies, which are liked by grown-ups. Most of the movies belong to the same cluster even though different information sources are used. However, there are some objects that fall into different clusters with respect to different sources. In this example, movie "Wall-E" by genre is liked by kids, but is liked by grown-ups based on the user watching history, and thus it is likely to be a horizontal anomaly. Finding such "inconsistent" movies can help film distributors better understand the expected audiences of different movies and make smarter marketing plans.

Some example scenarios of horizontal anomaly detection are listed as follows. 1) In social networks, detecting people who fall into different social communities with respect to different online social networks would be interesting for user behavior analysis; 2) In bioinformatics, inconsistencies across different gene-gene interaction similarity graphs derived from patients with and without a certain disease represent the genes which are critical to the disease; 3) For better business marketing, one wants to find out the person who bought quite different items compared with his peers in the same social community based on the two information sources drawn from user purchase history and friendship networks; and 4) Inconsistencies across multiple module interaction graphs derived from different versions of a software project can be used to assist programmers. Besides the examples discussed above, identifying horizontal anomalies across multiple sources can find applications in many other fields including smarter planet, internet of things, intelligent transportation systems, marketing, banking, etc.

To the best of our knowledge, this is the first work on identifying horizontal anomalies by exploring the inconsistencies among multiple sources. Most of the existing work on mining multiple information sources [188] concerns merging and synthesizing models, rules, patterns obtained from multiple sources by reconciling their differences, such as multi-view learning [37, 53, 151, 57] and multi-view clustering [21, 183, 121, 33]. As for multi-source anomaly detection, the studies focus on how to identify anomalies within a specific context [153, 165]. Although these studies take two types of attributes (behavioral and contextual [31]) into consideration, they cannot be generalized to horizontal anomaly detection spanning multiple information sources. The reason is that they simply detect anomalies from the behavioral attributes while the contextual attributes only provide the context in which the anomalies are detected. In some sense, these contextual anomalies are still extracted from one source, whereas the proposed method can identify objects with inconsistent behavior across multiple sources.

We assume that each individual information source captures some similarity relationships between objects that may be represented in the form of a similarity matrix. Note that although in the example shown in in Figure 6.2, the horizontal anomalies can be found by checking if its direct neighbors are different in the two graphs, this simple solution cannot work in practice. The reason is that the clustering structures are much more complicated and noisy in real problems, and thus a global method that can detect both the underlying clustering structure and horizontal anomalies is needed. Therefore, we propose a systematic approach to identify horizontal anomalies from multiple similarity matrices. A summary of this chapter is as follows:

Method. We combine the input matrices into one matrix that captures the information from each source, but also ensures that individual object relationships are preserved. We then adopt spectral techniques to identify the key eigenvectors of the graph Laplacian of the combined matrix, and identify horizontal anomalies by computing cosine distance between the components of these eigenvectors.

Interpretation. We give theoretical interpretations of the proposed method from both spectral clustering and random walk perspectives. The method can be regarded as conducting spectral clustering on multiple sources simultaneously with a joint constraint that their clusterings should be similar, and objects that are clustered differently are categorized as horizontal anomalies. The horizontal anomalies can also be regarded as those having long commute time in the random walk defined over the graph.

Experiments. We validate the proposed algorithm on both synthetic and real data sets, and the results demonstrate the advantages of the proposed approach in finding horizontal anomalies. For example, we find "One Flew Over the Cuckoo's Nest" and "Pulp Fiction" as the most anomalous, while "Star Wars" as the least anomalous among the top 20 popular movies from a set of 7595 movies.

6.2 Methodology

Suppose we have a set of N objects $X = \{x_1, x_2, \ldots, x_N\}$ and there are P information sources that describe different aspects of these objects. Let $W^{(t)}$ denote the similarity matrix derived from the t-th information source where the ij-th entry $w_{ij}^{(t)}$ represents the similarity between objects x_i and x_j $(i \neq j)$ with respect to the t-th source. Let $W_{ii}^{(t)} = 0$ for all t and i. The objective is to assign an anomalous score s_i to each object x_i, which represents how likely the object is anomalous when its behavior differs among the P different information sources. In the simple example shown in Figure 6.2, there are two matrices that describe pairwise similarities among the 7 objects, and we expect that x_4 will have the highest anomalous score.

In this section, we present a **HO**rizontal **A**nomaly **D**etection (**HOAD**) algorithm to solve the proposed problem. The basic idea is as follows: As discussed in Section 6.1, we assume that the available information sources on the same set of objects have similar clustering structures, and thus if an object is assigned to different clusters when using various information sources, it can be regarded as a horizontal anomaly. This suggests that we can first cluster the objects separately in each source and compare the clustering results. However, because clustering is unsupervised learning, we do not know the correspondence between clusters in different clustering solutions. We solve this problem by adding the constraint that the same object should be put into the same cluster by all the clustering solutions as often as possible. Another problem is that in reality, an object

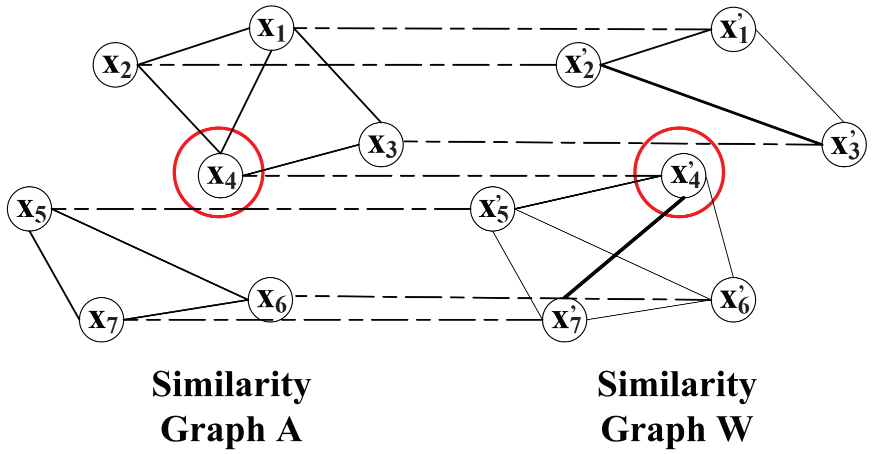


Figure 6.3: Illustration of Combined Graph for the Example in Figure 6.2

never belongs to just one cluster for sure, usually it can be assigned to several clusters with certain probabilities. Therefore, soft clustering is more desirable. In the proposed approach, we calculate the anomalous degree of an object based on how much its clustering solutions differ from each other. To simplify the notations, we start with the cases having two distinct information sources. We state the method in Section 6.2.1, give spectral clustering and random walk interpretations in Section 6.2.2, and explain how it is generalized to multiple information sources in Section 6.2.3.

6.2.1 HOAD Algorithm

Suppose we have N objects: $\{x_1, \ldots, x_N\}$ and two $N \times N$ similarity matrices on the objects: A and W, where a_{ij} and w_{ij} define the similarity between x_i and x_j from different aspects. The algorithm consists of two major steps: 1) Conduct soft clustering on A and W together with the constraint that an object should be assigned to the same cluster; 2) Quantify the difference between the two clustering solutions to derive anomalous scores.

The details are as follows. We start from constructing two similarity graphs from A and W. In each of them, each node denotes an object. If the similarity between two objects x_i and x_j is greater than 0, we connect an edge between x_i and x_j and the edge weight equals to the similarity between them. We construct a combined graph by connecting the nodes which correspond to the same object in the two graphs with an edge weighted m. m, a large positive number, is a penalty parameter. An example of such a graph is illustrated in Figure 6.3 for the toy example

Algorithm 4 HOAD algorithm

```
Input: similarity matrices $A$ and $W$, number of eigenvectors $k$, penalty parameter $m$,
Output: anomalous score vector $\vec{s}$;
Algorithm:
  Compute matrix $Z$ according to Eq. (6.1)
  Compute graph Laplacian $L$ as in Eq. (6.2)
  Conduct eigen-decomposition of $L$ and Let $H$ be the $k$ smallest eigenvectors with smallest eigenvalues
  Compute anomalous score of each object $s_i$ based on Eq. (6.4) and Eq. (6.5) for $i = 1, \ldots, N$
  return $\vec{s}$
```

shown in Figure 6.2. The set of nodes in the combined graph consists of two copies of the objects: $\{x_1, \ldots, x_N, x'_1, \ldots, x'_N\}$ ($2N$ nodes in total). Let M be an $N \times N$ diagonal matrix with m on the diagonal:

$$M = \text{diag}(m, m, \ldots, m).$$

Clearly, $M = m \cdot I$ where I is an $N \times N$ identify matrix and m represents the constraint put across the two information sources. Let Z be the adjacency matrix of the combined graph, which is a $2N \times 2N$ matrix:

$$Z = \begin{bmatrix} A\ M \\ \\ MW \end{bmatrix}. \tag{6.1}$$

We cluster the nodes in the combined graph. As can be seen, there are two copies of the objects in the combined graph and with the help of the edge between the copies of the same object, we cluster the objects in the same way across different sources. In Section 6.2.2, we give a theoretical justification of this claim. First, we compute the graph Laplacian L as:

$$L = D - Z \tag{6.2}$$

using degree matrix D (a $2N \times 2N$ diagonal matrix):

$$D = \text{diag}\big(\{\sum_{j=1}^{2N} z_{ij}\}_{i=1}^{2N}\big). \tag{6.3}$$

Secondly, compute the k smallest eigenvectors of L (with smallest eigenvalues) and let $H \in \mathbb{R}^{2N \times k}$ be the matrix containing these eigenvectors as columns. We divide H into two submatrices U and

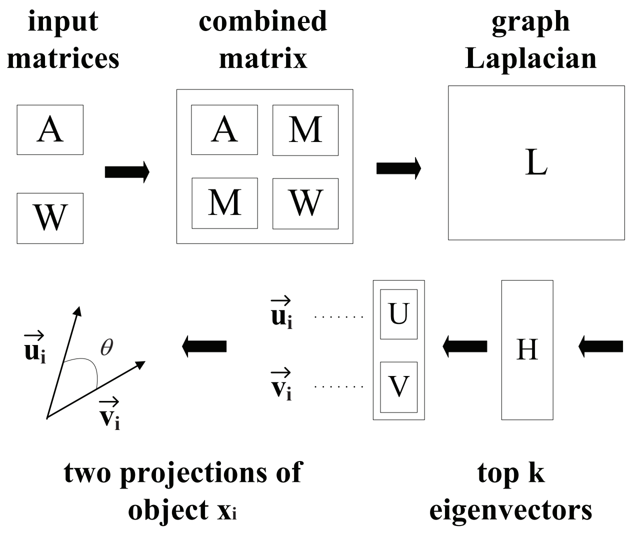


Figure 6.4: Algorithmic Flow of HOAD Algorithm

V each with size $N \times k$ so that $H = \begin{bmatrix} U \\ V \end{bmatrix}$. Therefore, the i-th and $(i+N)$-th rows of H are represented as:

$$\vec{u}_i = \vec{h}_i, \quad \vec{v}_i = \vec{h}_{i+N}, \tag{6.4}$$

which correspond to two "soft clustering" representations of x_i with respect to A and W respectively. Finally, compute the anomalous score for object x_i using cosine distance between the two vectors:

$$s_i = 1 - \frac{\vec{u}_i \cdot \vec{v}_i}{||\vec{u}_i|| \cdot ||\vec{v}_i||}. \tag{6.5}$$

The algorithm flow is summarized in both Algorithm 4 and Figure 6.4. We start with two $N \times N$ similarity matrices A and W, and combine them together with the penalty constraint matrix M to form a combined matrix Z. After that, we compute Z's graph Laplacian L and conduct eigen decomposition on L. H contains the k smallest eigenvectors of L as column vectors, and it is divided into two $N \times k$ submatrices U and V. For each $i \in \{1, \dots, N\}$, $\vec{u}_i$ and $\vec{v}_i$, i.e., the i-th rows of U and V, can be regarded as the two clustering results of x_i. We compute the anomalous score of x_i as the cosine distance between $\vec{u}_i$ and $\vec{v}_i$, which is $1 - \cos(\theta)$ with θ representing the angle between the two vectors.

6.2.2 Interpretations

In this part, we explain the algorithm from the perspectives of spectral clustering and random walk.

Clustering on Combined Graph. As can be seen, we first perform spectral clustering on the combined graph in Algorithm 4, but we replace the clustering step by anomalous score computation. Now we show that the algorithm can be interpreted as conducting constrained spectral clustering on the two similarity graphs simultaneously. The basic idea of spectral clustering is to project the objects into a low-dimensional space (defined by the k smallest eigenvectors of the graph Laplacian matrix) so that the objects in the new space can be easily separated. We call the projections as *spectral embeddings* of the objects. It has been shown that the matrix formed by the k eigen vectors (H) of L is the solution to the following optimization problem [125]:

$$min_{H\in\mathbb{R}^{N\times k}} \quad \mathrm{Tr}(H'LH) \quad \text{s.t.} \quad H'H = I \tag{6.6}$$

Since we define the graph Laplacian L as $D-Z$ (Eq. (6.2)), the objective function is thus equivalent to:

$$\begin{aligned} &min_{H\in\mathbb{R}^{2N\times k}} \mathrm{Tr}(H'DH) - \mathrm{Tr}(H'ZH) \\ &\text{s.t.} \quad H'H = I \end{aligned} \tag{6.7}$$

Let $f(H) = \mathrm{Tr}(H'ZH)$ and $g(H) = \mathrm{Tr}(H'DH)$. H is a $2N \times k$ matrix, and again we divide it into two submatrices U and V: $H = [U \quad V]^T$. Also, from the definition of Z (Eq. (6.1)), we have:

$$\begin{aligned} f(H) = &\mathrm{Tr}\left([U' \quad V'] \begin{bmatrix} A & M \\ M & W \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} \right) \\ =&\mathrm{Tr}(U'AU + V'WV + V'MU + U'MV) \\ =&\mathrm{Tr}(U'AU) + \mathrm{Tr}(V'WV) + 2m\sum_{i=1}^{N}\sum_{j=1}^{k} u_{ij}v_{ij} \end{aligned} \tag{6.8}$$

Suppose the degree matrices for A and W are D^a and D^w respectively:

$$D^a = \mathrm{diag}\big(\{\sum_{j=1}^{N} a_{ij}\}_{i=1}^{N}\big), \quad D^w = \mathrm{diag}\big(\{\sum_{j=1}^{N} w_{ij}\}_{i=1}^{N}\big).$$

The row sum of M is always m. Based on the definition of D (Eq. (6.3)), we have

$$\begin{aligned} g(H) =& \mathrm{Tr}\left([U' \quad V'] \left(\begin{bmatrix} D^a 0 & \\ 0 \quad D^w \end{bmatrix} + mI \right) \begin{bmatrix} U \\ V \end{bmatrix} \right) \\ =& \mathrm{Tr}(U'D^aU + V'D^wV + mU'U + mV'V) \end{aligned} \tag{6.9}$$

Also,

$$H'H = I \Leftrightarrow [U' \quad V'] \begin{bmatrix} U \\ V \end{bmatrix} = I \Leftrightarrow U'U + V'V = I$$

By putting $f(H)$ and $g(H)$ together and ignoring the constant term $m \cdot \mathrm{Tr}(U'U + V'V)$, we have an equivalent formulation of the problem in Eq. (6.6):

$$\begin{aligned} & min_{U,V \in \mathbb{R}^{N\times k}} \mathrm{Tr}(U'(D^a - A)U) + \mathrm{Tr}(V'(D^w - W)V) - 2m \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij} v_{ij} \\ & \text{s.t.} \quad U'U + V'V = I \end{aligned} \tag{6.10}$$

Clearly, each of the first two terms in Eq. (6.10) corresponds to the spectral clustering problem using A or W alone. The third term acts as the constraint that the two clustering solutions should be similar (cosine similarity). Different from spectral clustering, we didn't actually perform the clustering procedure. Therefore, our method can be regarded as embedding the objects into two eigenspaces with respect to the two information sources while putting the constraint that the two projections should be similar. The parameter m controls how much we impose the constraint.

Note that in Algorithm 4, the i-th row vector in U (the first N rows of H) and V (the last N rows of H) contain the projections of object x_i. Due to the principle of spectral clustering, if the spectral embeddings $\vec{u}_i$ and $\vec{v}_i$ are close to each other, the corresponding object x_i is more likely to be assigned to the same cluster with respect to two different sources. Therefore, the cosine

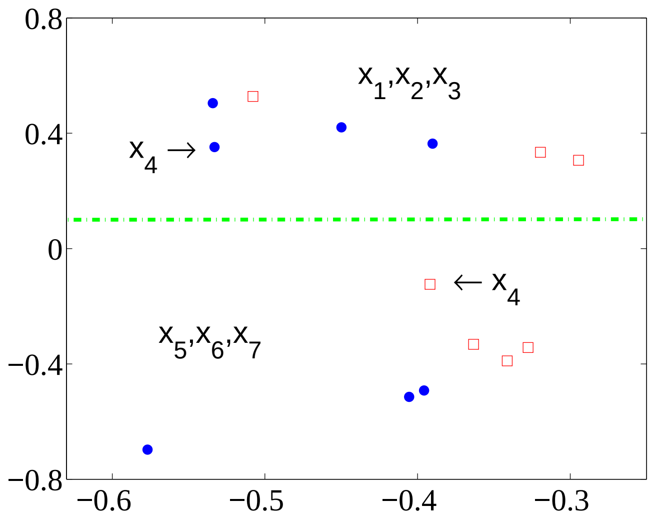


Figure 6.5: Two Smallest Eigenvectors for the Example in Figure 6.2

similarity between the two vectors $\vec{u}_i$ and $\vec{v}_i$ quantifies how similar the clustering results of object x_i on the two sources are, and thus represents its "normal" degree. In turn, the cosine distance as defined in Eq. (6.5) gives the "anomalous" degree of x_i with respect to the two sources. The higher the score s_i is, the more likely x_i is a horizontal anomaly.

Example. We show how the algorithm works through the example shown in Figure 6.3. After computing the graph Laplacian L of the combined graph's adjacency matrix according to Eq. (6.2), we extract the 2 smallest eigenvectors of L and put it into H, and thus H is a 14×2 matrix. The first seven rows correspond to the spectral embeddings of the seven objects with respect to the first source whereas the remaining ones are those with respect to the second source. In Figure 6.5, we plot these row vectors in a two-dimensional space where blue circles indicate the projections on the first source and red squares are results on the second source. Clearly, no matter which source we use, objects x_1, x_2 and x_3 are always projected on the top region of the space, whereas x_5, x_6 and x_7 are located at the bottom part. The biggest difference in the projections can be found in x_4, and thus it has the highest anomalous score among the seven objects.

Random Walk. In this part, we give some intuitions of the proposed method from the random walk perspective. Let z_{ij} be the edge weight between two nodes x_i and x_j in the graph. Let $d_i = \sum_{j=1}^{2N} z_{ij}$ be the degree of node x_i, and $\text{vol}(X) = \sum_{i=1}^{2N} d_i$ be the sum of all the edge weights in the graph. Suppose we define a random walk over the combined graph, where the transition probability from node x_i to node x_j is proportional to the edge weight in the graph: $p_{ij} = z_{ij}/d_i$. If the combined graph is connected and non-bipartite, then the random walk always has a unique invariant distribution $\pi = (\pi_1, \ldots, \pi_{2N})$, where $\pi_i = d_i/\mathrm{Vol}(X)$. Suppose x_i and x'_i are the two copies of the same object in the combined graph. Now we look at the commute distance between x_i and x'_i, which is the expected time it takes for the random walk to travel from x_i to x'_i and back. Instead of looking for one shortest path, the commute distance looks at all the possible paths. Therefore, only when there are many short paths from x_i to x'_i, their commute distance is small.

It is proven that commute distance on a graph can be computed with the help of the eigenvectors of the graph Laplacian L as defined in Eq. (6.2). Suppose L has eigenvalues $\lambda_1, \ldots, \lambda_{2N}$, and U and V are two $N \times N$ matrices containing all the eigenvectors for the two copies of the objects respectively. Let $\vec{u}_i$ and $\vec{v}_i$ denote the i-th row of U and V. We define $\vec{\gamma}$ as a length-$2N$ vector with each entry γ_l equal to $(\lambda_l)^{-0.5}$ if $\lambda_l \neq 0$, and 0 otherwise. Now we divide $\vec{\gamma}$ into two length-N vectors: $\vec{\gamma} = [\vec{\gamma}_u \quad \vec{\gamma}_v]$. Suppose we map x_i and x'_i into a new feature space where they are represented as $\vec{u}_i \cdot \vec{\gamma}_u$ and $\vec{v}_i \cdot \vec{\gamma}_v$ respectively. It can be derived that the commute distance c_i between x_i and x'_i is:

$$c_i = \mathrm{vol}(X)||\vec{u}_i \cdot \vec{\gamma}_u - \vec{v}_i \cdot \vec{\gamma}_v||^2,$$

which is the Euclidean distance between the nodes in the new feature space scaled by vol(X). Recall that we compute the anomalous score of x_i as $1 - \frac{\vec{u}_i \cdot \vec{v}_i}{||\vec{u}_i|| \cdot ||\vec{v}_i||}$. We can see that both the anomalous score and the commute distance can be represented as *a distance function applied on the spectral embeddings of the two copies of the object.* The difference is: 1) The embeddings are scaled by $(\lambda_l)^{-0.5}$ in the commute distance; 2) All the eigenvectors are used in the commute distance whereas only the k smallest are used in the anomalous score computation; and 3) Euclidean distance is used instead of cosine distance in the commute distance.

Although the connection is loose, commute distance can be a helpful intuition to understand the anomalous scores. If it takes longer time to commute between the two copies of object x_i in the graph, x_i is more likely to be a horizontal anomaly. By the definition of commute distance, it means that it is hard to find many paths to travel between them. In fact, in the combined graph, the only way to travel from the left side to the right side is through the constraint edge with weight

m. Therefore, a horizontal anomaly is the object that can be categorized into different clusters with respect to different information sources, which is consistent with our definition. As an example, it is hard to travel between x_4 and x'_4 in the graph shown in Figure 6.3 because they link to different sets of objects in the two sources, and thus x_4 is a horizontal anomaly. On the contrary, besides the constraint edge connecting x_1 and x'_1, x_1 can travel to x'_1 through many other paths because its neighbors in the cluster maintain the same in the two graphs. Therefore, its commute distance is small and x_1 is a normal object.

6.2.3 Multiple Sources

We can adapt Algorithm 4 to handle more than two information sources as follows. Suppose we have similarity matrices $\{W^{(1)}, W^{(2)}, \ldots, W^{(P)}\}$ as the input.

Graph Construction. The combined graph is constructed in a similar fashion as discussed before: Duplicate the objects for P copies, in each copy retain the similarity information from each source, and connect each pair of the nodes corresponding to the same object with an edge weighted m. The adjacency matrix Z is thus a $NP \times NP$ matrix with $\{W^{(1)}, W^{(2)}, \ldots, W^{(P)}\}$ on the diagonal and the diagonal matrix encoding constraints $M = \text{diag}(m, m, \ldots, m)$ on all the off-diagonal blocks. We can make the framework more flexible by allowing for different m values for different pairs of information sources. m is a user-provided parameter, which characterizes the similarity between information sources in their clustering structures. Therefore, one principle to set m is to assign a larger value to it if the two information sources are more likely to share the same clustering results. In the experiments, to reduce the number of parameters, we use the simple version where we set m a uniform value. However, how to set m is still a tricky problem because m can take any value between 0 and infinity. In Section 6.3, we give some discussions on how to transform the problem of setting m to an easier task.

Eigenvectors of Graph Laplacian. After Z is obtained, we calculate its graph Laplacian and the k smallest eigenvectors following exactly the same procedure as in Algorithm 4. One concern is that, when the number of information sources increases, the size of the matrix L grows quadratically and this leads to higher computation and storage cost. However, the graph Laplacian of Z is a matrix with special structures where most entries are 0, and also, we only need the k smallest eigenvectors instead of the full eigenspace. Therefore, this problem is much easier than the general eigen-decomposition problem on any matrix. In fact, efficient packages such as ARPACK [111], have been developed to compute a few eigenvectors of large-scale sparse matrix. In Section 6.3, we show that the proposed method implemented based on ARPACK can scale well even when there are more than two information sources. Furthermore, we can use some parallel computing frameworks to process large matrices. For example, large scale top k eigensolver is available [97] using highly scalable MapReduce framework[1]. Another practical issue is how to choose the appropriate k, i.e., the number of eigenvectors we extract from the combined matrix. Choosing k is a general problem for all clustering algorithms, and a variety of methods have been developed. In particular, eigengap heuristic is proposed to choose k such that the first to the k-th eigenvalues are very small, but the $(k+1)$-th is relatively large. This heuristic works for spectral clustering methods as justified by spectral theory and perturbation theory. A brief discussion about these methods can be found in [125]. In Section 6.3, we show how the performance of the proposed method varies with respect to the value of k.

Anomalous Score Computation. The anomalous score is defined based on the distance between two vectors in Eq. (6.5). With P information sources ($P > 2$), we should calculate the anomalous degree of an object x_i based on the following P vectors: $\{\vec{h}_i, \vec{h}_{i+N}, \vec{h}_{i+2N}, \dots, \vec{h}_{i+(P-1)N}\}$. There are various ways to define a distance measure among multiple vectors. In the experiment, we simply use the average pairwise distance as the measure:

$$s_i = \frac{1}{P(P-1)} \sum_{a=0}^{P-1} \sum_{b=0}^{P-1} \mathbb{1}_{a \neq b} \cdot \left[1 - \frac{\vec{h}_{i+aN} \cdot \vec{h}_{i+bN}}{||\vec{h}_{i+aN}|| \cdot ||\vec{h}_{i+bN}||} \right]$$

Similarity Computation. We need similarity matrices derived from multiple sources as input to the algorithm. The notion of "similarity" between objects varies with the types of information sources. In real practice, the set of objects can be represented in different incompatible formats by the available information sources. For example, webpages on the internet can be represented as bag of words feature vectors (webpage contents), or a huge graph (hyperlink relationships). We discuss how to compute the similarity matrix W for different data types as follows: 1) **Graph**: $w_{ij} = 1$ if there exists an edge connecting x_i and x_j and 0 otherwise. 2) **Continuous Data**: We

[1]http://mahout.apache.org/

can use a Gaussian kernel applied on Euclidean distance: $w_{ij} = \exp(-||x_i - x_j||^2/\sigma^2)$ where σ^2 is the parameter used in the kernel. 3) **Binary Data**: We can use Jaccard Index to compute the similarity: $w_{ij} = |x_i \cap x_j|/|x_i \cup x_j|$. 4) **Documents**: Each document x_i is usually represented as a bag-of-words vector. The cosine similarity between two documents x_i and x_j is defined as: $w_{ij} = (x_i \cdot x_j)/(||x_i|| \cdot ||x_j||)$. Note that these are simply some examples showing how the pairwise similarity can be computed. More discussions on the similarity computation can be found in [82].

6.3 Experiments

We evaluate the HOAD algorithm on synthetic data to show its detection accuracy, as well as real datasets including DBLP and MovieLens to validate its ability of identifying meaningful horizontal anomalies.

6.3.1 Synthetic Data

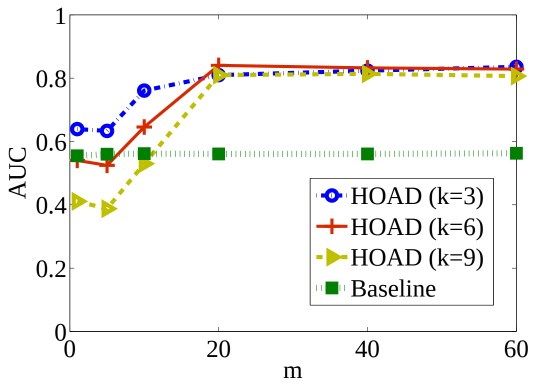
The concept of "horizontal anomaly" is new, and thus there are no benchmark datasets for it. Therefore, we propose a method to convert a classification problem into a horizontal anomaly detection problem, and then apply this procedure on several UCI machine learning data sets.

Data Generation. Recall that horizontal anomalies represent the objects that have inconsistent behavior among multiple information sources. Therefore, the basic idea of the transformation is to simulate "inconsistencies" by swapping feature values of objects from different classes. Suppose we have a training set from a classification problem where each object consists of feature values and a class label. Suppose there are N objects in the training set: $\{x_1, \ldots, x_N\}$, and the features X can be partitioned into two views. We assume that each of the feature sets is correlated with the class label, and thus objects within the same class share similar feature values in each feature set. Therefore, for two objects x_i and x_j from different classes, if their feature values are swapped in one view but remain unchanged in the other, they have "inconsistent" behavior among these two views, and thus represent horizontal anomalies. In this way, we generate r pairs of horizontal anomalies. The datasets we use all have continuous values, and thus we use Gaussian kernels to calculate the similarity. We apply the above method on four data sets obtained from UCI machine learning repository[2]: Zoo, Iris, Letter and Waveform. On each data set, we randomly split the feature set into two subsets with equal number of features. We repeat the transformation procedure 50 times and at each time, we generate a data set with around 10% anomalies. We evaluate the HOAD algorithm on the 50 data sets and report the average accuracy.

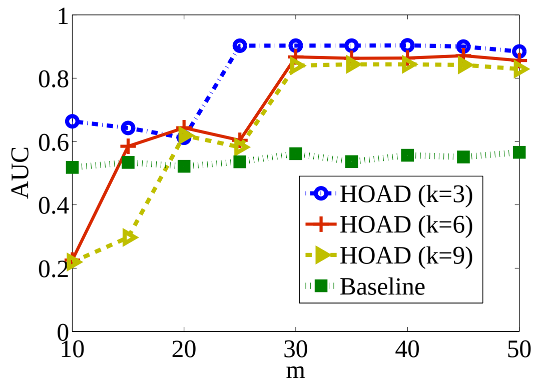
Evaluation Measure and Baseline Methods. For anomaly detection problems, one of the most widely used evaluation approaches is ROC analysis, which represents the trade-off between detection rate and false alarm rate. A good algorithm would produce an ROC curve as close to the left-top corner as possible, and thus the area under ROC curve (AUC), which is in the range [0,1], is a good evaluation metric. The higher the AUC is, the better the algorithm performs. We show the performance of the proposed HOAD algorithm with various parameter settings. The two most important parameters are the penalty value m and the number of eigenvectors k. Note that the proposed algorithm conducts a constrained soft clustering on multiple information sources simultaneously. Instead of conducting a joint clustering, the baseline method clusters the two sources *separately* and calculates the anomalous scores based on the difference between the two clustering solutions. Specifically, in two-source problems, we conduct eigen decomposition on the graph Laplacian matrices of the two similarity matrices A and W separately. Suppose the top k eigen representation of object x_i are u_i and v_i respectively, then we use Eq. (6.5) to compute the anomalous score of x_i for the baseline approach. Note that the major difference between the HOAD algorithm and the baseline method is on how to compute u_i and v_i.

Performance. The experimental results on the four data sets are shown in Figure 6.6(a) to Figure 6.6(d) where we vary the values of the parameters m and k. m indicates the penalty we enforce when the two clustering solutions do not agree, and k is the number of top eigenvectors. The baseline clusters the two sources separately, so neither m nor k is used in the baseline and its performance remains mostly stable except slight fluctuation due to random sampling in data generation. From the experimental results, we can see that HOAD algorithm generally outperforms the baseline, especially when k is small (e.g., $k = 3$). However, when the value of m is higher, the difference in AUC between the algorithms using different k is much smaller. Therefore, we focus on how to select the appropriate m in the following discussion. On these UCI datasets, when

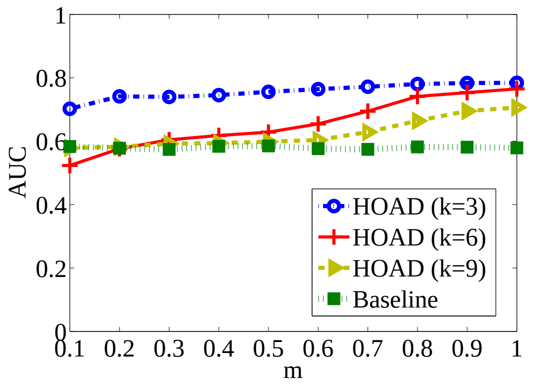
[2]http://archive.ics.uci.edu/ml



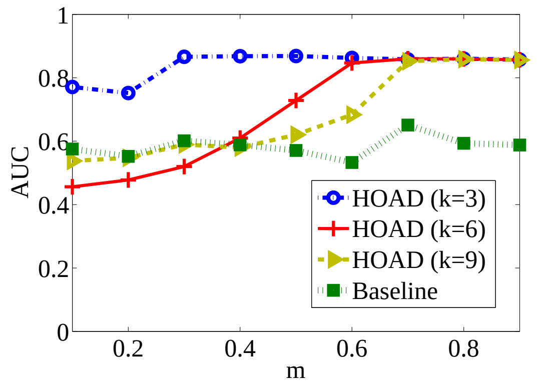
(a) Zoo



(b) Iris



(c) Letter



(d) Waveform

Figure 6.6: Anomaly Detection Performance on UCI Data

m increases, the proposed algorithm has a higher AUC. In the simulated study, the two feature sets are two disjoint subsets of the original features, and usually using all of the features lead to a better classification model. Hence the two views are correlated and using a large m captures this correlation well. However, this does not mean that we should assign a big number to m in all cases because this pattern may not always hold in real horizontal anomaly detection tasks. In the following experiments on DBLP data sets, we illustrate the relationship between m and the anomalous scores, and state some principles in setting m.

In Figure 6.7, we show the running time of HOAD algorithm with respect to 1000 to 6000 objects represented in two, three or four information sources. We conduct the experiments on synthetic data sets where we randomly generate similarity matrices for 50 trials, and report the average running time. The eigenvectors are computed using Matlab eigs function, which is based on ARPACK package. As can be seen, the HOAD algorithm can scale well to large data sets when

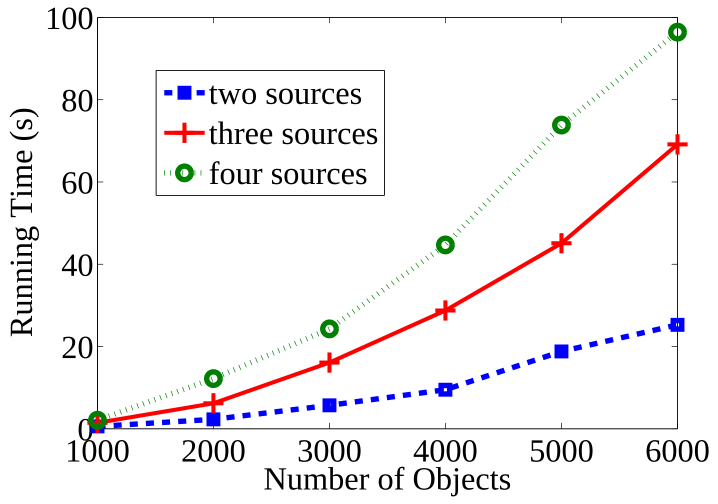


Figure 6.7: Running Time of HOAD Algorithm

the number of objects and number of sources both increase.

6.3.2 Real Case Studies

We discuss the issues of setting parameters on DBLP data and present illustrative results on MovieLens data.

DBLP. DBLP[3] provides bibliographic information on major computer science journals and proceedings. We define two horizontal anomaly detection tasks based on the DBLP data where the objects are a set of conferences and authors respectively. Let $N = 4220$, and the set of conferences is represented as $\{x_1, x_2, \ldots, x_N\}$. There are two views describing the conferences: the keywords in the conferences and the authors who published in the conferences. Specifically, each x_i has two vectors, each of which has the form $(x_{i1}, x_{i2}, \ldots, x_{iT})$. In the first vector, T is the total number of words, and x_{il} is the number of times the l-th word appeared in the i-th conference profile (we concatenate the titles of papers in the conference as the conference profile). In the second vector, T is the total number of authors in the second vector, and x_{il} denotes the number of times the l-th author published in the i-th conference. The pairwise similarity between two conferences x_i and x_j is defined as the cosine similarity between the corresponding vectors. Therefore, the conferences that share lots of keywords, or share lots of authors are similar. Similarly, we select a set of

[3]http://www.informatik.uni-trier.de/∼ley/db/

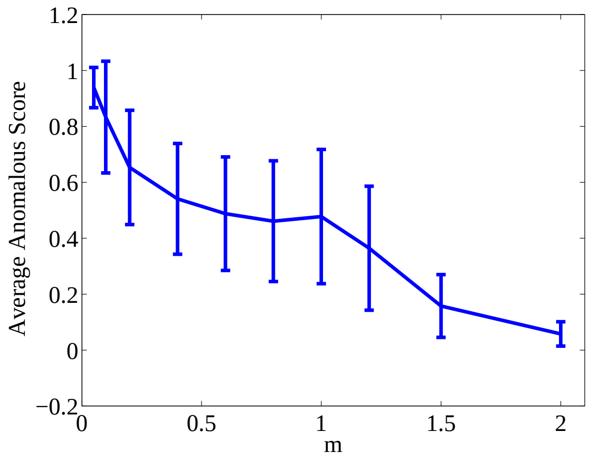


Figure 6.8: Parameter Study on Conferences

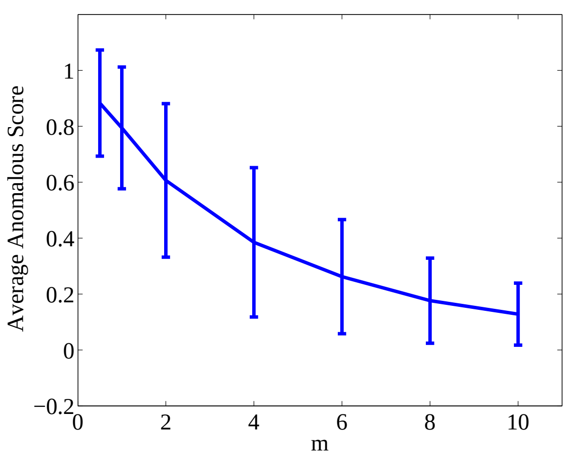


Figure 6.9: Parameter Study on Authors

3116 authors from data mining related areas and extract two types of information from DBLP: the publications and the co-authorships. Each author x_i is also represented in vectors $(x_{i1}, x_{i2}, \ldots, x_{iT})$ where in the first vector x_{il} denotes the occurrence of the l-th word in the authors' publications, and x_{il} corresponds to the number of times x_i and x_l collaborate in the second one. Cosine similarity is used, and similar authors will share co-authors, or keywords in their publications.

We study the effect of m on the anomalous scores. For each m, we apply the HOAD algorithm to the data sets, and compute the mean and standard deviation of the objects' anomalous scores. The results on conferences and authors are shown in Figure 6.8 and Figure 6.9 where the points on the line are the average anomalous scores and the error bar denotes the standard deviation. Obviously, the average anomalous score decreases as m increases. Recall that the anomalous scores indicate the degree of differences between the spectral embeddings derived from the two similarity matrices. When we give a heavy penalty on different embeddings by the two sources, we basically bias the two projections towards the ones that agree the most. Therefore, when m is larger, the spectral embeddings from the two sources are more likely to be the same, and thus the difference between them is smaller. On the contrary, when m is small, the constraint on the similarities between the two projections is often violated, so most of the objects are projected differently.

Another observation is that the variance among the anomalous scores goes up first and then goes down as m increases. When m is quite large or quite small, the two projections of all the objects would be very similar or very different, and thus the objects receive similar anomalous scores. There exist a large variability among the anomalous scores only when m is in the middle of the spectrum. Although m can be drawn from $(0, \infty)$, the average anomalous scores are within a fixed range–[0,1]. Usually, the two projections are positively correlated, and thus their cosine similarity is between 0 and 1. Therefore, we can choose m which leads to an average anomalous score around 0.5 because the variance of the anomalous scores usually reaches the highest point here and this helps us identify the horizontal anomalies.

MovieLens. We use the Movielens dataset[4] with movies as objects and three sources of information to capture their relationships: 1) Genre Information: Individual movies are classified as being of one or more of 18 genres, such as Comedy and Thriller, which can be treated as binary vectors. 2) User Viewing Information: Individual movies have a list of users that watched the movie. This may also be represented as a vector (per movie) across all users. 3) User Tagging Information: Individual movies are tagged by different users. Looking across all users, we can determine a vector per movie. In all three cases, we compute the pairwise similarity using cosine similarity across the vectors. The data set contains 10 million ratings and 100,000 tags for 10681 movies by 71567 users. We choose a set of 7595 movies, each of which has both ratings and tags. We then have three similarity matrices, corresponding to these three different sources for all these movies, and use our techniques to identify horizontal anomalies. To evaluate the performance of the HOAD algorithm on MovieLens dataset, we label some movies as "horizontal anomalies" based on the list of weirdest movies[5]. There are 572 movies listed as weirdest movies and among them 127 are found in the MovieLens dataset. These 127 movies are labeled as "anomalous" and the remaining 7468 movies are "normal". Based on these labels, we calculate the area under ROC curve (AUC) of both HOAD and the baseline method based on their computed anomalies scores for the 7595 movies. HOAD algorithm achieves a better AUC (0.4879) compared with that of the baseline method (0.4423). This demonstrates the ability of the proposed HOAD algorithm in detecting inconsistent movies across various information sources.

Moreover, we present the anomalous scores for the 20 most popular movies[6] as shown in Table 6.1. As may be seen, the movies "One Flew Over the Cuckoo's Nest" and "Pulp Fiction" are

[4]http://www.grouplens.org/node/73

[5]http://366weirdmovies.com/the-weird-movie-list

[6]As listed on http://www.imdb.com/chart/top on November 2010.

Table 6.1: Anomalous Scores of 20 Popular Movies from MovieLens

Movie	Score	Movie	Score
One Flew Over the Cuckoo's Nest	0.8079	Seven Samurai	0.6404
Pulp Fiction	0.7713	Fight Club	0.6364
Casablanca	0.7205	City of God	0.6278
The Shawshank Redemption	0.6949	The Lord of the Rings: The Return of the King	0.3512
The Godfather: Part II	0.6822	The Lord of the Rings: The Fellowship of the Ring	0.3478
The Godfather	0.6770	The Good, the Bad and the Ugly	0.3194
Goodfellas	0.6768	Raiders of the Lost Ark	0.3181
Schindler's List	0.6755	Rear Window	0.3095
12 Angry Men	0.6713	Star Wars	0.2982
The Dark Knight	0.6535	Star Wars: Episode V-The Empire Strikes Back	0.2562

identified as horizontal anomalies, as they tend to show strong disagreement between the genre classification, and the sets of users that watched and tagged them. Intuitively, this is expected as these two movies do not really fit into one classification or user category. Borrowing reviews from Wikipedia[7], "Pulp Fiction" is known for its rich, eclectic dialogue, ironic mix of humor and violence, and nonlinear storyline, which make it different and anomalous among movies. For "One Flew Over the Cuckoo's Nest", the review says "it is a comedy that can't quite support its tragic conclusion". These tell us the reasons why these two movies are detected as being inconsistent. On the other hand, "Star Wars" receives the lowest anomalous score as it attracts a particular set of audiences.

6.4 Related Work

Spectral clustering [125] is an effective clustering technique that has shown its advantages in many real-world applications. Some constrained spectral clustering algorithms [119, 166] have been proposed to incorporate pairwise must-link and cannot-link constraints. Different from our study, these algorithms take one information source and pairwise constraints on some of the objects as input. Consensus clustering [155, 54, 138] or multi-view clustering [21, 183, 121, 33] approaches try to compute a globally optimal clustering solution from multiple clustering solutions or multiple views. Usually the final solution represents the consensus among multiple clusterings. Although the proposed HOAD algorithm has a spectral clustering interpretation, our goal is to identify the *disagreement* between sources rather than reach a *consensus* among them. Spectral methods have been used in detecting online changes in time-dependent networks [85] or single-source anomalies

[7]http://en.wikipedia.org

[100], but these methods cannot be used to detect multi-source inconsistencies. The work in [90] compares two graphs to detect anomalies where an adaptive neighborhood selection procedure based on sparse graphical Gaussian model is proposed. However, they focus on finding correlation anomalies among *attributes* instead of *objects*.

6.5 Summary

We propose to detect horizontal anomalies, or objects that have inconsistent behavior among multiple sources. Intuitively, they belong to different clusters when considering many aspects from multiple information sources. Potential applications of the proposed approach are in cyber-security, social networking and internet of things. The proposed algorithm has two intrinsic steps. In the first step, we construct a combined similarity graph based on the input similarity matrices and compute the k smallest eigenvectors of the graph Laplacian as spectral embeddings of the objects. After that, we calculate the anomalous score of each object as the cosine distance between different spectral embeddings. The physical meaning of the proposed algorithm is explained from both constrained spectral clustering and random walk point of view. Experimental results show that the proposed HOAD algorithm can consistently find horizontal anomalies from DBLP and MovieLens datasets, where other anomaly detection methods fail to identify.

Chapter 7

Inconsistency Detection for Information Networks

As shown in Chapter 6, we can benefit from inconsistency detection across sources when unusual behavior is detected by comparing different sources. Although the proposed framework can be applied to a variety of applications, there are some specific learning scenarios that require more focused solutions. In this chapter and the following chapter, we present our study of inconsistency detection on two specific cases with multiple heterogeneous sources.

In this chapter, we consider a network where each node denotes an object and each link represents connections between two objects. We call such networks as *information networks*. Closely related objects that share the same properties or interests form a community in the network. By comparing node and link information, we can identify outliers (anomalies) within the context of communities such that the identified outliers deviate significantly from the rest of the community members. To automatically detect such outliers, we propose an probabilistic model, which formulates networked data as a mixture model composed of multiple normal communities and randomly generated outliers, characterizing both data and links simultaneously [69]. The model parameters and outliers are inferred by maximizing the posterior probability. This algorithm can be applied to a variety of networked systems, such as biological networks, social networks and traffic networks.

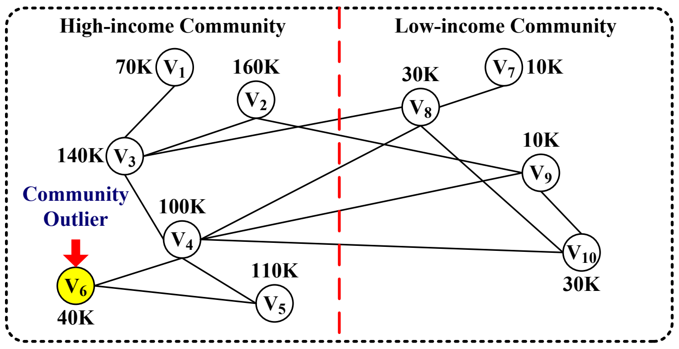
7.1 Overview

The problem of anomaly detection has been widely studied [31] in the scenario of one data source, where anomalies are defined as the objects that have deviant behavior compared with majority of the data. In many scenarios, however, an object may only be considered abnormal in a specific context but not globally [153, 165]. Such contextual outliers are sometimes more interesting and important than global outliers. For example, 20 Fahrenheit degree is not a global outlier in temperature, but it represents anomalous weather in the spring of New York City.

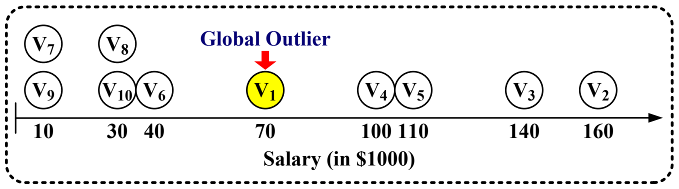
We study the problem of finding contextual outliers in an "information network". Networks have been used to describe numerous physical systems in our everyday life, including Internet composed of gigantic networks of webpages, friendship networks obtained from social web sites, and co-author networks drawn from bibliographic data. We regard each node in a network as an object, and there usually exist large amounts of information describing each object, e.g. the hypertext document of each webpage, the profile of each user, and the publications of each researcher. The most important and interesting aspect of these datasets is the presence of links or relationships among objects, which is different from the feature vector data type that we are more familiar with. We refer to the networks having information from both objects and links as information networks. Intuitively, objects connected via the network have many interactions, subsequently share mutual interests, and thus form a variety of communities in the network [76]. For example, in a blogsphere, there could be financial, literature, and technology cliches. Taking communities as contexts, we aim at detecting outliers that have non-conforming patterns compared with other members in the same community.

Example: Low-income person with rich friends

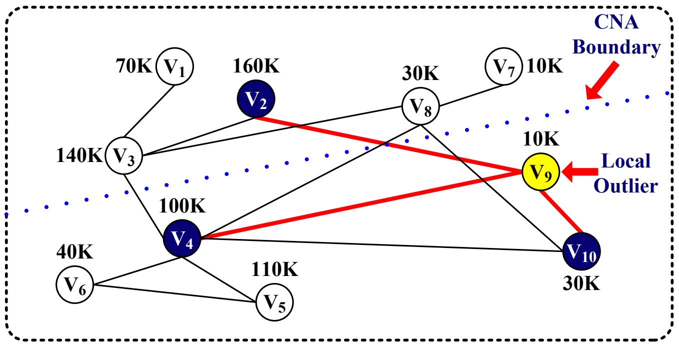
A friend network is shown in Figure 7.1(a), where each node denotes a person, and a link represents the friend relationship between two persons. Each person's annual salary is shown as a number attached to each node. There obviously exist two communities, high-income (v_1,v_2,v_3,v_4,v_5) and low-income (v_7,v_8,v_9,v_{10}). Interestingly, v_6 is an example of community outliers. It is only linked to the high-income community (70 to 160K), but has a relatively low income (40K). This person could be a rising star in the social network, for example, a young and promising entrepreneur, or someone who may settle down in a rich neighborhood. Another example is a co-author network. Researchers are linked through co-authorship, and texts are extracted from publications for each author in bibliographic databases. A researcher who does independent research on a rare topic is an outlier among people in his research community, for example, a linguistic researcher in the area of data mining. Additionally, an actor cooperation network can be drawn from movie databases where actors are connected if they co-star a movie. Exceptions can be found when an actor's profile



(a) Community Outliers–CODA



(b) Global Outliers–GLODA



(c) Local Outliers-DNODA

Figure 7.1: Comparison of Different Types of Outliers

deviates much from his co-star communities, such as a comedy actor co-starring with lots of action movie stars.

Limitation of Traditional Approaches

Identifying community outliers is a non-trivial task. First, if we conduct outlier detection only based on each object's information, without taking network structure into account, the identified outliers would only be "global" outliers. As shown in Figure 7.1(b), v_1 is a global outlier with 70K deviating from the other salary amounts in the "low-income person with rich friends" example. We call this method GLobal Outlier Detection Algorithm (**GLODA**). Secondly, when only "local" information (i.e., information from neighboring nodes) is considered, the identified node is just significantly away from its adjacent neighbors. It is a “local” outlier, not necessarily a “community” outlier. As illustrated in Figure 7.1(c), v_9 is a local outlier because his salary is quite different from those of his direct friends (v_2, v_4 and v_{10}). The corresponding algorithm is denoted as Direct Neighbor Outlier Detection Algorithm (**DNODA**).

In detecting community outliers, both the information at each individual object and the one in the network should be taken into account simultaneously. A naive solution is to first partition the network into several communities using network information [149, 99], and then within each community, identify outliers based on the object information. This two-stage algorithm is referred to as Community Neighbor Algorithm (**CNA**). The problem with such a two-stage approach is that communities discovered using merely network information may not make much sense. For example, partitioning the graph in Figure 7.1(c) along the dotted line minimizes the number of normalized cuts, and thus the line represents the boundary between two communities identified by CNA. However, the resulting two communities have wide-spread income levels, and thus it does not make much sense to detect outliers in such two communities.

Therefore, we propose to utilize both the network and data information in an integrated solution, in order to improve community discovery and find more meaningful outliers. The algorithm we developed is called community outlier detection algorithm (**CODA**). With the proposed method, the network in Figure 7.1(a) will be divided by the dashed line, and v_6 is detected as the community outlier. In many applications, no matter the network is dense or sparse, there is ambiguity in community partitions. This is particularly true for very large networks, since information from both nodes and links can be noisy and incomplete. Consolidating information from both sources can compensate missing or incomplete information from one side alone and is likely to yield a better solution.

Some clustering methods (**CLA** for short) have been developed to group nodes in an information network into communities using both data and link information [114, 175, 167]. Those methods, however, are not designed for outlier detection. The reason is that they are proposed under the assumption that there are no outliers. It is well-known that outliers can highly affect the formation of communities. Different from those methods, the proposed approach combines, instead

Table 7.1: Summary of Related Work

Algorithms	Tasks	Information Sources
GLODA	**global** outlier detection	data of objects
DNODA	**local** outlier detection	data and direct neighbors
CNA	find communities then detect outliers	use data and links **separately**
CLA	clustering in information networks	use data and links **together**

of separating, outlier detection and community mining into a unified framework. As summarized in Table 7.1, both GLODA and DNODA only use part of the available information, whereas the other two approaches consider both data and links. However, CNA utilizes the two information sources separately, and CLA is used to conduct clustering, instead of outlier detection.

Summary of the Proposed Approach

We propose a probabilistic model for community outlier detection in information networks. It provides a unified framework for outlier detection and community discovery, integrating information from both the objects and the network. The information collected at each object is formulated as a multivariate data point, generated by a mixture model. We use K components to describe normal community behavior and one component for outliers. Distributions for community components are, but not limited to, either Gaussian (continuous data) or multinomial (text data), whereas the outlier component is drawn from a uniform distribution. The mixture model induces a hidden variable z_i at each object node, which indicates its community. Then inference on z_i's becomes the key in detecting community outliers. We regard the network information as a graph describing the dependency relationships among objects. The links from the network (i.e., the graph) are incorporated into our modeling via a hidden Markov random field (HMRF) on the hidden variable z_i's. We motivate an objective function from the posterior energy of the HMRF model, and find its local minimum by using an Iterated Conditional Modes (ICM) algorithm. We also provide some methods for setting the hyper-parameters in the model. Moreover, the proposed model can be easily generalized to handle a variety of data as long as a distance function is defined.

A summary of our contributions is as follows:

- Finding community outliers is an important problem but has not received enough attention

Table 7.2: Important Notations

Symbol	Definition
$I = \{1, \ldots, i, \ldots, M\}$	the indices of objects
$V = \{v_1, \ldots, v_M\}$	the set of objects
$S = \{s_1, \ldots, s_M\}$	the given attribute values of the objects
$W_{M \times M} = [w_{ij}]$	the given link structure, w_{ij}-the link strength between objects v_i and v_j
$Z = \{z_1, \ldots, z_M\}$	the set of random variables for hidden labels of the objects
$X = \{x_1, \ldots, x_M\}$	the set of random variables for observed data
$N_i \quad (i \in I)$	the neighborhood of object v_i
$1, \ldots, k, \ldots, K$	the indices of normal communities
$\Theta = \{\theta_1, \ldots, \theta_K\}$	the set of random variables for model parameters
$\theta_k = \{\mu_k, \sigma_k^2\}$	the parameters of the k-th normal community (continuous data): μ_k-mean, σ_k^2-variance
$\theta_k = \{\beta_{k1}, \beta_{k2}, \ldots, \beta_{kT}\}$	the parameters of the k-th normal community (text data)
$\beta_{kl} \quad (l = 1, \ldots, T)$	the probability of observing the l-th word in the k-th community (text data)

in the field of information network analysis. To the best of our knowledge, this is the first work on identifying community outliers by analyzing both the data and links simultaneously.

- We propose an integrated probabilistic model to interpret normal objects and outliers, where the object information is described by some generative mixture model, and network information is encoded as spatial constraints on the hidden variables via a HMRF model.

- Efficient algorithms based on EM and ICM algorithms are provided to fit the HMRF model as well as inferring the hidden label of each object.

- We validate the proposed algorithm on both synthetic and real data sets, and the results demonstrate the advantages of the proposed approach in finding community outliers.

7.2 Community Outlier Detection

Community outliers can be defined in various ways. We define it based on a generative model unifying data and links. Based on the definition, we discuss the specific models for continuous data and text data. Table 7.2 summarizes some important notations.

7.2.1 Outlier Detection via HMRF

The problem is defined as follows: suppose we have an information network denoted as a graph $G = (V, W)$, where V denotes a set of objects $\{v_1, \ldots, v_M\}$, and W represents the links between each pair of objects. Specifically, the input include:

- $S = \{s_1, \ldots, s_M\}$ where s_i is the data associated with object v_i.
- W is the symmetric $M \times M$ adjacency matrix of the network where w_{ij} ($w_{ij} \geq 0$) is the weight of the link between the two objects v_i and v_j. If $w_{ij} > 0$, v_i and v_j are connected.

Let $I = \{1, \ldots, M\}$ be the set of indices of the M objects. The objective is to derive the anomalous subset $\{i : v_i \text{ is a contextual outlier with respect to } S \text{ and } W,\ i \in I\}$.

Next, we discuss how to formulate this using HMRF model. Mathematically, a HMRF model is characterized by the following:

Observed data

$X = \{x_1, \ldots, x_M\}$ is a set of random variables. Each random variable x_i generates the data s_i associated with the i-th object.

Hidden labels

$Z = \{z_1, \ldots, z_M\}$ is the set of hidden random variables, whose values are unobservable. Each variable z_i indicates the community assignment of v_i. Suppose there are K communities, then $z_i \in \{0, 1, \ldots, K\}$. If $z_i = 0$, v_i is an outlier. If $z_i = k$ ($k \neq 0$), v_i belongs to the k-th community.

Neighborhood system

The links in W induce dependency relationships among the hidden labels, with the rationale that if two objects v_i and v_j are linked on the network (i.e., they are neighbors), then they are more likely to belong to the same community (i.e., z_i and z_j are likely to have the same value). However, since outliers are randomly generated, the neighbors of an outlier are not necessarily outliers. So we adjust the neighborhood system as the following:

$$N_i = \begin{cases} \{j; w_{ij} > 0, i \neq j, z_j \neq 0\} & z_i \neq 0 \\ \phi & z_i = 0. \end{cases}$$

Here N_i stands for the set of neighbors of object v_i. When $z_i \neq 0$, i.e., v_i is not an outlier, the neighborhood of v_i contains its normal neighbors in G. In contrast, v_i's neighborhood is empty if it is an outlier ($z_i = 0$).

Conditional independence

The set of random variables X are conditionally independent given their labels:

$$P(X = S|Z) = \prod_{i=1}^{M} P(x_i = s_i|z_i).$$

Normal Communities and Outliers

We assume that the k-th normal community ($k \neq 0$) is characterized by a set of parameters θ_k, i.e.,

$$P(x_i = s_i|z_i = k) = P(x_i = s_i|\theta_k).$$

Quite differently, the outliers follow a uniform distribution, i.e.,

$$P(x_i = s_i|z_i = 0) = \rho_0$$

where ρ_0 is a constant. Let $\Theta = \{\theta_1, \ldots, \theta_K\}$ be the set of all parameters describing the normal communities.

Dependency between hidden variables

The random field defined over the hidden variables Z is a Markov random field, where the Markov property is satisfied:

$$P(z_i|z_{I-\{i\}}) = P(z_i|z_{N_i}) \quad z_i \neq 0.$$

It indicates that the probability distribution of z_i depends only on the labels of v_i's neighbors in G if z_i corresponds to a normal community. If $z_i = 0$, v_i is an outlier and is not linked to any other objects in the random field, and thus we set $P(z_i = 0) = \pi_0$ where π_0 is a constant. According to the Hammerskey-Clifford theorem [17], an MRF can equivalently be characterized by a Gibbs distribution:

$$P(Z) = \frac{1}{H_1} \exp(-U(Z)) \tag{7.1}$$

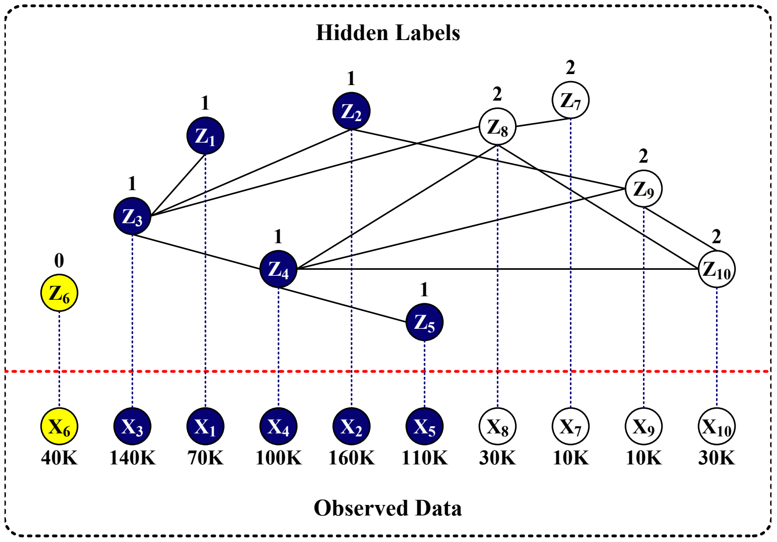


Figure 7.2: Community Outlier Detection Model

where H_1 is a normalizing constant, and $U(Z) = \sum_{c \in C} V_c(Z)$, the potential function, is a sum of clique potentials $V_c(Z)$ over all possible cliques ($c \in C$) in G. Since outliers are stand-alone objects (their links in G are ignored in the model), we define the potential function only on the neighborhood of normal objects:

$$U(Z) = -\lambda \sum_{w_{ij}>0, z_i \neq 0, z_j \neq 0} w_{ij} \delta(z_i - z_j) \tag{7.2}$$

where λ is a constant, $w_{ij} > 0$ denotes that there is a link connecting the two objects v_i and v_j, and both z_i and z_j are non-zero. The δ function is defined as $\delta(x) = 1$ if $x = 0$ and $\delta(x) = 0$ otherwise. The potential function suggests that, if v_i and v_j are normal objects, they are more likely to be in the same community when there exists a link connecting them in G, and the probability becomes higher if their link w_{ij} is stronger.

Figure 7.2 shows the HMRF model for the example in Figure 7.1(a). The top layer represents the hidden variables $\{z_1, \ldots, z_{10}\}$. It has the same topology as the original network G except that the neighborhood of z_6 is now empty because it is an outlier. Given $z_i = k$, the corresponding data value is generated according to the parameter θ_k. The bottom layer is composed of the data values (salaries) of the objects. In this example, two communities are formed, and objects in the same community are strongly linked in the top layer, as well as having similar values in the bottom layer. When considering both data and link information, we cannot assign v_6 to any community (linked to community 1 but its value is closer to community 2), and thus regard it as a community outlier.

7.2.2 Modeling Continuous and Text Data

In the proposed model, the probability of hidden variables is modeled by Eq. (7.1) and Eq. (7.2), and the outliers are generated by a uniform distribution. However, given the hidden variable $z_i \neq 0$, the probability distribution of x_i can be modeled in various ways depending on the format it is taking. In this part, we discuss how $P(x_i = s_i|z_i)$ $(z_i \neq 0)$ is modeled when s_i is continuous or a text document, the two major types of data we encounter in applications. Extensions to general cases are discussed in Section 7.4.

Continuous Data

For the sake of simplicity, we assume that the data S are 1-dimensional real numbers. Extensions to model multi-dimensional continuous data are straightforward. We propose to model the normal points in S by a Gaussian mixture due to its flexibility in approximating a wide range of continuous distributions. Parameters needed to describe the k-th community are the mean μ_k and variance σ_k^2: $\theta_k = \{\mu_k, \sigma_k^2\}$. Given the model parameter $\Theta = (\theta_1, \ldots, \theta_K)$, if $z_i = k \in \{1, \ldots, K\}$, the logarithm of the conditional likelihood $\ln P(x_i = s_i|z_i = k)$ is:

$$\ln P(x_i = s_i|z_i = k) = -\frac{(s_i - \mu_k)^2}{2\sigma_k^2} - \ln \sigma_k - \ln \sqrt{2\pi}. \tag{7.3}$$

Text Data

Suppose each object v_i is a document that is comprised of a bag of words. Let $\{w_1, w_2, \ldots, w_T\}$ be all the words in the vocabulary, and each document is represented by a vector $s_i = (d_{i1}, d_{i2}, \ldots, d_{iT})$, where d_{il} denotes the count of word w_l in v_i. Now the parameter characterizing each normal community is $\theta_k = \{\beta_{k1}, \beta_{k2}, \ldots, \beta_{kT}\}$ where $\beta_{kl} = P(w_l|z_i = k)$ is the probability of seeing word w_l in the k-th community. Given that a document v_i is in the k-th community, its word counts s_i follow a multinomial distribution, and thus $\ln P(x_i = s_i|z_i = k)$ is defined as:

$$\ln P(x_i = s_i|z_i = k) = \sum_{l=1}^{T} d_{il} \ln P(w_l|z_i = k) = \sum_{l=1}^{T} d_{il} \ln \beta_{kl}. \tag{7.4}$$

7.3 Fitting Community Outlier Detection Model

In the HMRF model for outlier detection we discussed in Section 7.2, both the model parameters Θ and the set of hidden labels Z are unknown. In this section, we present the method to infer the values of hidden variables (Section 7.3.1) and estimate model parameters (Section 7.3.2).

7.3.1 Inference

We first assume that the model parameters in Θ are known, and discuss how to obtain an assignment of the hidden variables. The objective is to find the configuration that maximizes the posterior distribution given Θ. We then discuss how to estimate Θ and Z simultaneously in Section 7.3.2.

In general, we seek a labeling of the objects, $Z = \{z_1, \ldots, z_M\}$, to maximize the posterior probability (MAP):

$$\hat{Z} = \arg\max_{Z} P(X = S|Z)P(Z).$$

We use the Iterated Conditional Modes (ICM) algorithm [18] to solve this MAP estimation problem. It adopts a greedy strategy by calculating local minimization iteratively and the convergence is guaranteed after a few iterations. The basic idea is to sequentially update the label of each object, keeping the labels of the other objects fixed. At each step, the algorithm updates z_i given $x_i = s_i$ and the other labels by maximizing $P(z_i|x_i = s_i, z_{I-\{i\}})$, the conditional posterior probability. Next we discuss the two scenarios separately when z_i takes non-zero or zero values.

If $z_i \neq 0$, we have

$$P(z_i|x_i = s_i, z_{I-\{i\}}) \propto P(x_i = s_i|Z)P(Z).$$

As discussed in Eq. (7.1) and Eq. (7.2), the probability distribution of Z is given by

$$P(Z) \propto \exp\Big(\lambda \sum_{w_{ij}>0, z_i \neq 0, z_j \neq 0} w_{ij}\delta(z_i - z_j)\Big).$$

In $P(z_i|x_i = s_i, z_{I-\{i\}})$, the links that involve objects other than v_i are irrelevant, and thus

$$P(z_i|x_i = s_i, z_{I-\{i\}}) \propto P(x_i = s_i|z_i) \cdot \exp\Big(\lambda \sum_{j \in N_i} w_{ij}\delta(z_i - z_j)\Big)$$

where only the links between v_i and its neighbors in N_i are taken into account. We take logarithm of the posterior probability, and then transform the MAP estimation problem to the minimization of the conditional posterior energy function:

$$U_i(k) = -\ln P(x_i = s_i|z_i = k) - \lambda \sum_{j \in N_i} w_{ij}\delta(k - z_j).$$

If $z_i = 0$, v_i has no neighbors, and thus

$$P(z_i|x_i = s_i, z_{I-\{i\}}) \propto P(x_i = s_i|z_i = 0)P(z_i = 0) = \exp(-U_i(0)) \tag{7.5}$$

with

$$U_i(0) = -\ln(\rho_0\pi_0) = a_0.$$

Therefore, to find z_i that maximizes $P(z_i|x_i = s_i, z_{I-\{i\}})$, it is equivalent to minimizing the posterior energy function: $\hat{z}_i = \arg\min_k U_i(k)$ where

$$U_i(k) = \begin{cases} -\ln P(x_i = s_i|z_i = k) - \lambda \sum_{j \in N_i} w_{ij}\delta(k - z_j) k \neq 0 \\ a_0 \qquad k = 0 \end{cases} \tag{7.6}$$

As can be seen, λ is a predefined hyper-parameter that represents the importance of the network structure. $\ln P(x_i = s_i|z_i = k)$ is defined in Eq. (7.3) and Eq. (7.4) for continuous and text data respectively. To minimize $U_i(k)$, we first select a normal cluster k^* such that $k^* = \arg\min_k U_i(k)(k \neq 0)$. Then we compare $U_i(k^*)$ with $U_i(0)$, which is a predefined threshold a_0. If $U_i(k^*) > a_0$, we set $\hat{z}_i = 0$, otherwise $\hat{z}_i = k^*$. As shown in Algorithm 5, we first initialize the label assignment for all the objects, and then repeat the update procedure until convergence. At each run, the labels are updated sequentially by minimizing $U_i(k)$, which is the posterior energy given $x_i = s_i$ and the labels of the remaining objects.

Algorithm 5 Updating Labels

Input: set of data S, adjacency matrix W, set of model parameters Θ, number of clusters K, link importance λ, threshold a_0, initial assignment of labels $Z^{(1)}$;
Output: updated assignment of labels Z;
Algorithm:

```
Randomly set $Z^{(0)}$
$t \leftarrow 1$
while $Z^{(t)}$ is not close enough to $Z^{(t-1)}$ do
    $t \leftarrow t + 1$
    for $i = 1; i <= M; i++$ do
        update $z_i^{(t)} = k$ which minimizes $U_i(k)$ in Eq. (7.6).
    end for
end while
return $Z^{(t)}$
```

7.3.2 Parameter Estimation

In Section 7.3.1, we assume that Θ is known, which is usually unrealistic. In this part, we consider the problem of estimating unknown Θ from the data. Θ describes the model that generates S, and thus we seek to maximize the data likelihood $P(X = S|\Theta)$ to obtain $\hat{\Theta}$. However, because both the hidden labels and the parameters are unknown and they are inter-dependent, it is intractable to directly maximize the data likelihood. We view it as an "incomplete-data" problem, and use the expectation-maximization (EM) algorithm to solve it.

The basic idea is as follows. We start with an initial estimate $\Theta^{(0)}$, then at E-step, calculate the conditional expectation $Q(\Theta|\Theta^{(t)}) = \sum_Z P(Z|X, \Theta^{(t)}) \ln P(X, Z|\Theta)$, and at M-step, maximize $Q(\Theta|\Theta^{(t)})$ to get $\Theta^{(t+1)}$ and repeat. In the HMRF outlier detection model, we can factorize $P(X, Z|\Theta)$ as $P(X|Z, \Theta)P(Z)$, and since $P(Z)$ is not related to Θ, we can regard the corresponding terms as a constant in Q. Similarly, the outlier component does not contribute to estimation of Θ neither, and thus $\sum_{i=1}^{n} P(z_i = 0|x_i = s_i) \ln P(x_i = s_i|z_i = 0)$ can also be absorbed into the constant term H_2:

$$Q = \sum_{i=1}^{M} \left(\sum_{k=1}^{K} P(z_i = k|x_i = s_i, \Theta^{(t)}) \ln P(x_i = s_i|z_i = k, \Theta) \right) + H_2. \tag{7.7}$$

We approximate $P(z_i = k|x_i = s_i, \Theta^{(t)})$ using the estimates obtained from Algorithm 5, where $P(z_i = k^*|x_i = s_i, \Theta^{(t)}) = 1$ if $k^* = \arg\min_k U_i(k)$, and 0 otherwise.

Specifically, for continuous data, we maximize Q to get the mean and variance of each normal

Algorithm 6 Community Outlier Detection

```
Input: set of data $S$, adjacency matrix $W$, number of clusters $K$, link importance $\lambda$, threshold $a_0$;
Output: set of outliers;
Algorithm:
  Initialize $Z^0$,$Z^1$ randomly
  $t \leftarrow 1$
  while $Z^{(t)}$ is not close enough to $Z^{(t-1)}$ do
    M-step: Given $Z^{(t)}$, update the model parameters $\Theta^{(t+1)}$ according to Eq. (7.8) and Eq. (7.9) (continuous
    data), or Eq. (7.10) (text data).
    E-step: Given $\Theta^{(t+1)}$, update the hidden labels as $Z^{(t+1)}$ using Algorithm 5.
    $t \leftarrow t + 1$
  end while
  return the indices of outliers: $\{i : z_i^{(t)} = 0, i \in I\}$
```

community $k \in \{1, \ldots, K\}$, where $\ln P(x_i = s_i|z_i = k, \Theta)$ is defined in Eq. (7.3):

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{M} P(z_i = k|x_i = s_i, \Theta^{(t)})s_i}{\sum_{i=1}^{M} P(z_i = k|x_i = s_i, \Theta^{(t)})}, \tag{7.8}$$

$$(\sigma_k^{(t+1)})^2 = \frac{\sum_{i=1}^{M} P(z_i = k|x_i = s_i, \Theta^{(t)})(s_i - \mu_k)^2}{\sum_{i=1}^{M} P(z_i = k|x_i = s_i, \Theta^{(t)})}. \tag{7.9}$$

Similarly, for text data, based on Eq. (7.4), as well as the constraints that $\sum_{l=1}^{T} \beta_{kl} = 1$ $(k = 1, \ldots, K)$, we have:

$$\beta_{kl}^{(t+1)} = \frac{\sum_{i=1}^{M} P(z_i = k|x_i = s_i, \Theta^{(t)})d_{il}}{\sum_{l=1}^{T}\sum_{i=1}^{M} P(z_i = k|x_i = s_i, \Theta^{(t)})d_{il}} \tag{7.10}$$

for $k = 1, \ldots, K$ and $l = 1, \ldots, T$.

In summary, the community outlier detection algorithm works as follows. As shown in Algorithm 6, we begin with some initial label assignment of the objects. In the M-step, the model parameters are estimated by maximizing the Q function based on the current label assignment. In the E-step, we run Algorithm 5 to re-assign the labels to the objects by minimizing $U_i(k)$ for each node v_i sequentially. The E-step and M-step are repeated until convergence is achieved, and thus the outliers are the nodes that have 0 as the estimated labels. Note that the running time is linear in the number of edges. It is not worse than any baseline that uses links because each edge has to be visited at least once. For dense graphs, the time can be quadratic in the number of objects. However, in practice, we usually encounter sparse graphs, on which the method runs in linear time and can scale well.

7.4 Discussions

To use the community outlier detection algorithm more effectively, the following questions need to be addressed: 1) How to set the hyper parameters? 2) What is a good initialization of the label assignment Z? 3) Can the algorithm be applied to any type of data?

Setting Hyper-parameters

We need users' input on three hyper-parameters: threshold a_0, link importance λ, and the number of components K. Intuitively, a_0 controls the percentage of outliers r discovered by the algorithm. We will expect a large number of outliers if a_0 is low and few outliers if a_0 is high. Therefore, we can transform the problem of setting a_0, which is difficult, to an easier problem to choose the percentage of outliers r. To do this, in Algorithm 5, we first let $\hat{z_i} = \arg\min_k U_i(k)(k \neq 0)$ for each $i \in I$, and sort $U_i(\hat{z_i})$ for $i = 1, \ldots, M$ and select the top r percent as outliers.

$\lambda > 0$ represents our confidence in the network structure where we put more weights on the network and less weights on the data if λ is set higher. Therefore, if λ is lower, the outliers found by Algorithm 6 is more similar to the results of detecting outliers merely based on nodes information. On the other hand, a higher λ makes the network structure play a more important role in community discovery and outlier detection. It is obvious that if we set λ to be extremely high, and the graph is connected, then every node will turn out to have the same label. To avoid such cases, we set an upper bound λ^C so that for any $\lambda > \lambda^C$, the results contain empty communities. With this requirement, we show that the proposed algorithm is not sensitive to λ in Section 7.5.

K is a positive integer, denoting the number of normal communities. In principle, it controls the scale of the community, and thus a small K leads to "global" outliers, whereas the outliers are determined locally if lots of communities are formed (i.e., large K). Many techniques have been proposed to set K effectively, for example, Bayesian information criterion (BIC), Akaike information criteria (AIC) and minimum description length (MDL). Here, we use AIC to set the number of normal communities. It is defined as:

$$AIC(\Delta) = 2b - 2 \ln P(X|\Delta) \tag{7.11}$$

where Δ denotes the set of hyper-parameters and b is the number of parameters to be estimated (model complexity). Since $P(X|\Delta)$ is hard to obtain, in the proposed algorithm, we use $P(X|\hat{Z}, \Delta)$ to approximate it by assuming that $\hat{Z}$ is the true configuration of the hidden variables.

Initialization

Good initialization is essential for the success of the proposed community outlier detection algorithm, otherwise the algorithm can get stuck at some local maximum. Instead of starting with a random initialization, we initialize Z by clustering the objects without assigning any outliers. Although this may affect the estimation of the model parameters at the first iteration, it can gradually get corrected while we update Z and nominate outliers in the E-step. To overcome the barrier of local maximum, we repeat the algorithm multiple times with different initialization and choose the one that maximizes the likelihood.

Extensions

We have provided models for continuous and text data, which already covers lots of applications. Here, we discuss extension of the proposed approach to more general data formats by using a distance function. In general, we let the center of each community μ_k to be the parameter characterizing the community, and define $D(s_i, \mu_k)$ to be the distance in feature values from any object s_i to the center of the k-th community μ_k. For $k = 1, \ldots, K$, we then define $P(x_i = s_i|z_i = k)$ in terms of the distance function:

$$P(x_i = s_i|z_i = k) \propto \exp(-D(s_i, \mu_k))$$

which suggests that given v_i is from a normal community, the probability of $x_i = s_i$ increases as the distance from s_i to μ_k gets closer. For example, we can choose D to be a distance function from the class of Bregman divergence [8], which is a generalization from the Euclidean distance and is known to have a strong connection to exponential families of distributions.

7.5 Experiments

The evaluation of community outlier detection itself is an open problem due to lack of groundtruths for community outliers. Therefore, we conduct experiments on synthetic data to compare detection accuracy with the baseline methods, and evaluate on real datasets to validate that the proposed algorithm can detect community outliers effectively.

7.5.1 Synthetic Data

In this part, we describe the experimental setting and results on the synthetic data.

Data Generation

We generate networked data through two steps. First, we generate synthetic graphs, which follow the properties of real networks–they are usually sparse, follow power law's degree distributions, and consist of several communities. The links of each object follow Zipf's law, i.e., most of the nodes have very few links, and only a few nodes connect to many nodes. We forbid self-links and remove the nodes that have no links. Secondly, we infer the label of each node following the proposed generative model, and sample a continuous number based on the label of each node. The configuration parameters to describe $P(X|Z)$ include the number of communities K and the percentage of outliers r. We draw the mean of each community uniformly from [-10,10], let the standard deviation be $10/K$, and generate random numbers using Gaussian probability density.

Baseline Methods

As discussed in Section 7.1, we compare the proposed community outlier detection algorithm (**CODA**) with the following outlier detection methods:

- **GLODA**: This baseline looks at the data values only. We use the popular outlier detection algorithm LOF [27] to detect "global" outliers without taking the network structure into account.

- **DNODA**: This method only considers the values of each object's direct neighbors in the

Table 7.3: Comparison of Precisions on Synthetic Data

Precisions		$K=5$				$K=8$			
		GLODA	DNODA	CNA	CODA	GLODA	DNODA	CNA	CODA
$M=1000$	$r=1\%$	0.0143	0.0714	0.5429	**0.6286**	0.0571	0.0571	0.4429	**0.7429**
	$r=5\%$	0.0867	0.2600	0.6930	**0.8106**	0.0688	0.1554	0.5723	**0.6565**
$M=2000$	$r=1\%$	0.0118	0.0111	0.1007	**0.6565**	0.0395	0.0170	0.1536	**0.4974**
	$r=5\%$	0.0567	0.1779	0.4645	**0.6799**	0.0649	0.1341	0.4944	**0.7047**
$M=5000$	$r=1\%$	0.0061	0.0041	0.0510	**0.3714**	0.0163	0.0000	0.0204	**0.5347**
	$r=5\%$	0.0496	0.1134	0.1854	**0.7302**	0.0565	0.0646	0.1602	**0.7926**

graph. We define the outlier score as:

$$\frac{\sum_{j\in N_i} D(s_i, s_j)}{|N_i|} \tag{7.12}$$

where D is the Euclidean distance function. N_i contains all the direct neighbors of v_i in the graph: $N_i = \{j : w_{ij} > 0, i \neq j\}$. If s_i is significantly different from the data of v_i's direct neighbors, it is considered an outlier.

- **CNA**: In this approach, we partition the graph into K communities using clustering algorithms [98], and define outliers as the objects that have significantly different values compared with the other objects in the same community. Therefore, the outlier score is calculated in the same way as in Eq. (7.12). But here, N_i stands for the whole community: $N_i = \{j : z_i = z_j, i \neq j\}$ where z_i is the community label derived from the clustering of the network structure.

Empirical Results

In experimental studies, we make each baseline method detect the same number of outliers as that of the groundtruths. To achieve this, we simply sort the outlier scores obtained by the three baseline methods in descending order, and take the top r percent as outliers. Then we use **precision**, also known as **true positive rate**, as the evaluation metric. It is defined as the percentage of correct ones in the set of outliers identified by the algorithm. We vary the scale of the network to have 1000, 2000 and 5000 nodes respectively. We set the number of clusters K to be either 5 or 8, and the percentage of outliers r to be either 1% or 5%.

For each parameter setting, we randomly generate 10 sets of networked data, and report the

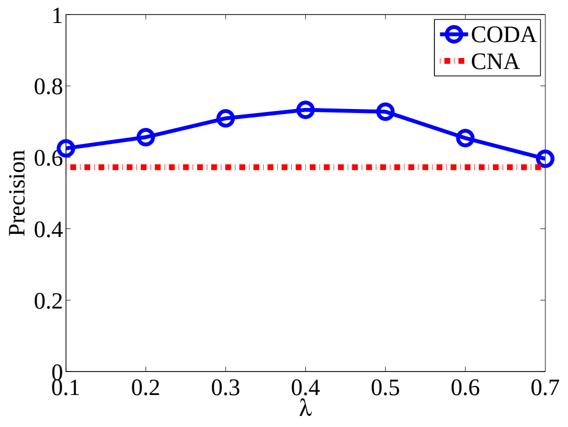


Figure 7.3: Parameter Sensitivity Analysis

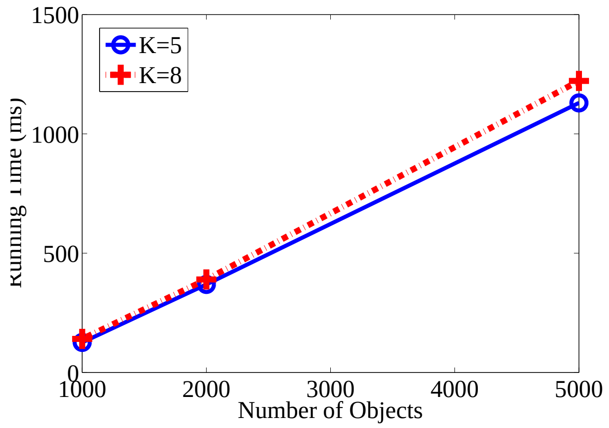


Figure 7.4: Running Time

average precisions of all the methods in Table 7.3. It is clear that GLODA fails to find most of community outliers because the method completely ignores the network structure information. The approach that only checks the direct neighbors of each object to determine outliers (DNODA) also has a low precision. On the other hand, if we first discover the communities, and then identify outliers based on the peers in the community, the precision is improved as shown in the method CNA. The proposed CODA algorithm further increases the precision by modeling both data and link information. We can observe the consistent improvements where the margin of precision increase is from 8% to 60%.

Sensitivity

Figure 7.3 shows the performance of the CODA algorithm when we vary λ from 0.1 to 0.7, as illustrated using the solid line. The dotted line represents the performance of the best baseline method CNA applied on the same data set. The results are obtained on the synthetic data with 1000 objects, 5 communities and 1% community outliers. It is clear that in spite of slight changes caused by parameter variation, the proposed method improves over the best baseline method. We let $\lambda = 0.2$ to get the experimental results shown in Table 7.3.

Time Complexity

Suppose the number of objects is M, and the number of edges is E. In M-step, we need to visit all the objects to calculate the model parameters, so the time complexity is $O(M)$. In E-step, for each object v_i, the posterior energy function U_i has to aggregate the effect of the labels of v_i's neighbors to compute $P(Z)$. Therefore, in principle, the time of the E-step is $O(E)$. Real network is usually sparse, and thus the computation time of the proposed approach can be linear in the number of objects. Figure 7.4 presents the average running time of the CODA algorithm on the synthetic data. We generate sparse networks using power law distribution where the number of edges grow linearly, and thus the running time is linear in the number of objects.

7.5.2 DBLP

DBLP[1] provides bibliographic information on major computer science journals and proceedings. We extract two sub-networks from the DBLP data: a conference relationship network and a co-authorship network.

Sub-network of Conferences

In the conference relationship network, we use 20 conferences from four research areas as the nodes of the graph, and construct a similarity graph based on the 20 nodes. Suppose there are L authors, then each conference has a $L \times 1$ vector A_i, whose l-th entry is the number of times the l-th author publishes in the i-th conference. We use cosine similarity to represent the link weight between two conferences:

$$w_{ij} = cos(A_i, A_j) = \frac{A_i \cdot A_j}{||A_i||||A_j||}. \tag{7.13}$$

This suggests that the conferences that attract the same set of authors have strong connections, and such conferences may form a research community. Additionally, we have a document attached to each node, which contains all the published titles in the conference. We conduct the community outlier detection algorithm on this network to obtain the outlier that has a different research theme compared with the other conferences containing similar researchers.

From this dataset, we find the following communities:

- **Database:** ICDE, VLDB, SIGMOD, PODS, EDBT
- **Artificial Intelligence:** IJCAI, AAAI, ICML, ECML

[1]http://www.informatik.uni-trier.de/∼ley/db/

Table 7.4: Top Words in Communities

Communities	Keywords
Data Mining	frequent dimensional spatial association similarity pattern fast sets approximate series
Database	oriented views applications querying design access schema control integration sql
Artificial Intelligence	reasoning planning logic representation recognition solving problem reinforcement programming theory
Information Analysis	relevance feature ranking automatic documents probabilistic extraction user study classifiers

- **Data Mining:** KDD, PAKDD, ICDM, PKDD, SDM
- **Information Analysis:** SIGIR, WWW, ECIR, WSDM

The community outliers detected by the proposed algorithm include **CVPR** and **CIKM**. Clearly, CVPR is more likely to fall into the AI area because researchers in CVPR will often attend IJCAI, AAAI, ICML and ECML. However, although people in computer vision utilize many general artificial intelligence methods, there exist unique computer vision techniques, such as segmentation, object tracking, and image modeling. Therefore, CVPR represents a community outlier in this problem. On the other hand, CIKM has a wide-spread scope, and attracts people from information analysis, data mining, and database areas. Apparently, it has a different research theme from that of any conference in these areas, and thus represents a community outlier as well.

<u>Sub-network of Authors</u>

We extract a co-authorship network, which contains the authors publishing in the 20 conferences mentioned above from DBLP. We select the top 3116 authors with the highest number of publications in these conferences, and use them as nodes of the network[2]. If two researchers have co-authored papers, there is an edge connecting them in the graph. The weight of the edge is the number of times two researchers have collaborated. We run the CODA algorithm on this co-author network to identify communities and community outliers. The top-10 frequent words occurring in each community identified by the algorithm are shown in Table 7.4. It is obvious that we can discover four research communities in this co-author network: Database (**DB**), Artificial Intelligence

[2]This is a sub-network of the original DBLP network. There could have some information loss in the co-authorship relationships.

Researchers & Collaborators	Research Interests
Dennis Shasha **DB** 19 **DM** 6	biological computing, pattern recognition, querying in trees and graphs, pattern discovery in time series, cryptographic file systems, database tuning
Craig A. Knoblock **IA** 4 **AI** 4 **DM** 1 **DB** 1	planning, machine learning, constraint reasoning, semantic web, information extraction, gathering, integration, mediators, wrappers, source modeling, record linkage, mashup construction, geospatial and biological data integration
Eric Horvitz **IA** 9 **AI** 4	human decision making, computational models of reflection, action with applications in time-critical decision making, scientific exploration, information retrieval, and healthcare
Sourav S. Bhowmick **IA** 8 **DM** 2 **DB** 2	blogs, social media analysis; web evolution, evolution, graph mining; social networks, XML storage, query processing, usability of XML/graph databases, indexing and querying graphs, predictive modeling, comparison of molecular networks, multi-target drug therapy
Timothy W. Finin **IA** 6 **AI** 1	social media, the semantic web, intelligent agents, pervasive computing
Jack Mostow **AI** 3 **IA** 2	focuses on using computers to listen to children read aloud while other interests include machine learning, automated replay of design plans, and discovery of search heuristics
Terrance E. Boult **AI** 2 **IA** 1	vision and security including video surveillance systems, biometrics, biometric fusion, supporting trauma treatment, steganalysis, network security, detection of chemical and biological weapons
Jayant R. Kalagnanam **DB** 3 **AI** 2 **IA** 1	decision support, optimization, economics and their applications to electronic commerce
Ken Barker **IA** 2 **AI** 2 **DB** 1	knowledge representation and reasoning, knowledge acquisition, natural language processing
Dimitris Achlioptas **AI** 4	threshold phenomena in random graphs and random formulas, applications of embeddings and spectral techniques in machine learning, algorithmic analysis of massive networks

Figure 7.5: Community Outliers in DBLP Co-authors

(**AI**), Data Mining (**DM**), and Information Analysis (**IA**).

Outliers in this sub-network somehow represent researchers who are conducting research on some different topics from his collaborators and peer researchers in the community. To illustrate the effectiveness of the proposed algorithm, we check the research interests listed on the homepages of the researchers identified by the CODA algorithm. In Figure 7.5, we show each researcher's name together with the number of his collaborators in each of the four communities (DB, AI, DM, and IA) in the first column. Their research interests are shown in the second column. As can be seen, these researchers indeed studied something different from his collaborators and the majority of the communities. For example, Jayant R. Kalagnanam mainly focuses on electronic commerce, which is a less popular topic among his collaborators in Database, Artificial intelligence and Information Analysis areas. Jack Mostow has focused on using computers to listen to children read aloud, which is a less studied research theme in Artificial Intelligence and Information Analysis. Through this example, we demonstrate that the proposed CODA algorithm has the ability of detecting outliers that deviate from the rest of the community.

7.6 Related Work

Outlier detection, sometimes referred to as anomaly or novelty detection, has received considerable attention in the field of data mining [31]. Outlier detection in data without considering contexts is called *global outlier detection*. Recently, people began to study how to identify anomalies within a specific context. These methods are able to detect interesting outliers or anomalies which cannot be found by existing outlier detection algorithms from a global view. Specifically, the pre-defined contextual attributes include spatial attributes [148, 158], neighborhoods in graphs [157], and some contextual attributes [153]. When there is no a priori contextual information available, Wang et al. propose to simultaneously explore contexts and contextual outliers based on random walks [165]. The proposed community outlier problem differs from these papers in that we use communities in networks as contexts, and they are inferred based on both *data* and *link* information.

Outliers identified in network structures purely by link analysis is referred to as *structural outliers* [174]. There are also works devoting to finding unusual sub-graph patterns in networks [133]. Clearly, these types of outliers are not the same as the community outliers we defined. In general, outlier detection is *unsupervised*, i.e., the task is to identify something novel or anomalous without the aid of labeled data. There exist some semi-supervised outlier detection approaches that take labeled examples as prior knowledge of label distribution [179, 160, 58]. Different from these methods, we aim at *unsupervised* outlier detection on networked data requiring no labeled data.

In recent years, many methods have been developed to discover clusters or communities in networks [76]. At first, community discovery is conducted on links only without consulting objects' information. Such techniques find communities as strongly connected sub-networks by similarity computation [102, 93] or graph partitioning [149, 125, 99]. Later, it was found that utilizing both link and data information leads to the discovery of more accurate and meaningful communities [114, 175, 167]. Some relational clustering methods [70, 123] fall into this category when both attributes of objects and relationships between objects are considered. Among various techniques, Markov random field [115, 181] is commonly used to model the structural dependency among random variables and has been successfully applied to many applications, such as image segmentation. More generally, relational learning explores use of link structure in inference and learning problems

[73]. Moreover, some semi-supervised clustering techniques based on must-links and cannot-links [9] can be used to discover communities on networked data as well, where network structures provide must-links. As shown in the experiments, separating community discovery and outlier detection cannot work as well as our unified model because absorbing outliers into normal communities affect the profiling of normal communities, and in turn degrade the performance of the second stage outlier detection.

7.7 Summary

In this chapter, we discuss a new outlier detection problem in networks containing rich information, including data about each object and relationships among objects. We detect outliers within the context of communities such that the identified outliers deviate significantly from the rest of the community members. We propose a generative model called CODA that unifies both community discovery and outlier detection in a probabilistic formulation based on hidden Markov random fields. We assume that normal objects form K communities and outliers are randomly generated. The data attributes associated with each object are modeled using mixture of Gaussian distributions or multinomial distributions, whereas links are used to calculate prior distributions over hidden labels. We present efficient algorithms based on ICM and EM techniques to learn model parameters and infer the hidden labels of the community outlier detection model. Experimental results show that the proposed CODA algorithm consistently outperforms the baseline methods on synthetic data, and also identifies meaningful community outliers from the DBLP network data.

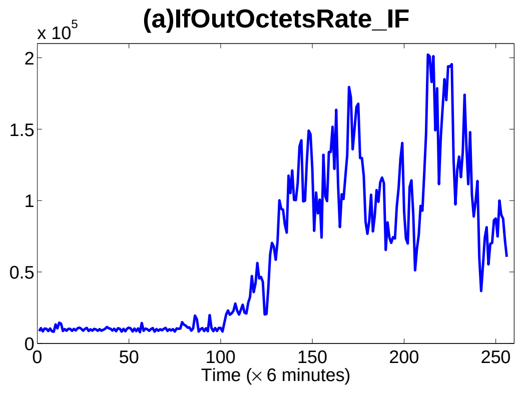
Chapter 8

Inconsistency Detection for System Debugging

In today's large-scale distributed systems, it is important to detect anomalous system behavior from the large amount of system monitoring data. This procedure is usually referred to as *System Debugging*. A distributed system consists of multiple connected machines, and monitoring data collected from each machine can be regarded as an information source. Although some knowledge about system problems can be extracted from each individual information source, a much richer body of knowledge can only be obtained by exploring the correlations or interactions across different sources. Specifically, the correlations between measurements collected across the distributed system can be used to infer normal system behavior, and thus a reasonable model to describe such correlations is crucially important in detecting and locating system problems. We propose a transition probability model to characterize pairwise measurement correlations [66]. Different from existing methods, the proposed solution can discover both the spatial (across system measurements) and temporal (across observation time) correlations, and thus such a model can successfully represent the system normal profiles. Whenever a record cannot be explained by the correlation model, it represents an anomaly. The effectiveness of this framework is demonstrated in its ability of detecting anomalous events and locating problematic sources from real monitoring data of three companies' infrastructures.

8.1 Overview

Recent years have witnessed the rapid growth of complexity in large-scale information systems. For example, the systems underlying Internet services are integrated with thousands of machines, and thus possess unprecedented capacity to process large volume of transactions. Therefore, large amount of system measurements (metrics) can be collected from software log files, system audit



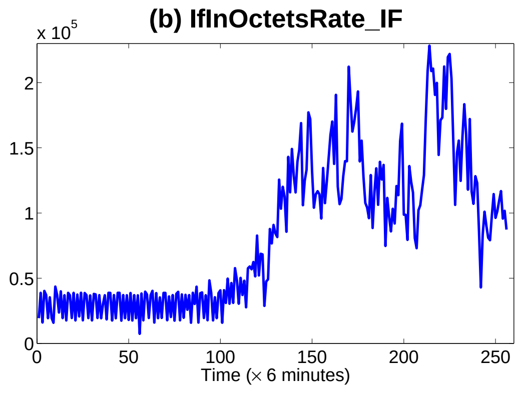


Figure 8.1: Measurements as Time Series.

events and network traffic statistics. To provide reliable services, system administrators have to monitor and track the operational status of their infrastructures in real time and fix any problems quickly. Due to the scale and complexity of the system, we have to automate the problem determination process so as to reduce the Mean Time to Recovery (MTTR). It is a challenging task to automatically detect anomalies in a large system because both the normal and anomalous behavior are heterogeneous and dynamic. In fact, the widely existing correlations among measurements are very useful for autonomic system management. Therefore, we propose a novel method that can effectively characterize the correlations across different system measurements and observation time. The method captures the complicated and changing normal profiles, and thus can be used to quickly detect and locate system problems.

Each distributed system usually consists of thousands of components, such as operating systems, databases, and application softwares. On each component, we are interested in its usage parameters, such as CPU and memory utilization, free disk space, I/O throughput and so on. Suppose we monitor l measurements for a particular system, and each measurement m^a $(1 \leq a \leq l)$ is uniquely defined by the component (e.g., database) and the metric (e.g., memory usage). Due to the dynamic nature of workloads received by the system, the measurement values usually change with time. Therefore, each measurement m^a can be viewed as a time series. We call the set of time series collected from the system as the monitoring data. Correlations are commonly found among the measurements because some outside factors, such as work loads and number of user requests, may affect them simultaneously. For example, the two measurements shown in Figure 8.1 are correlated.

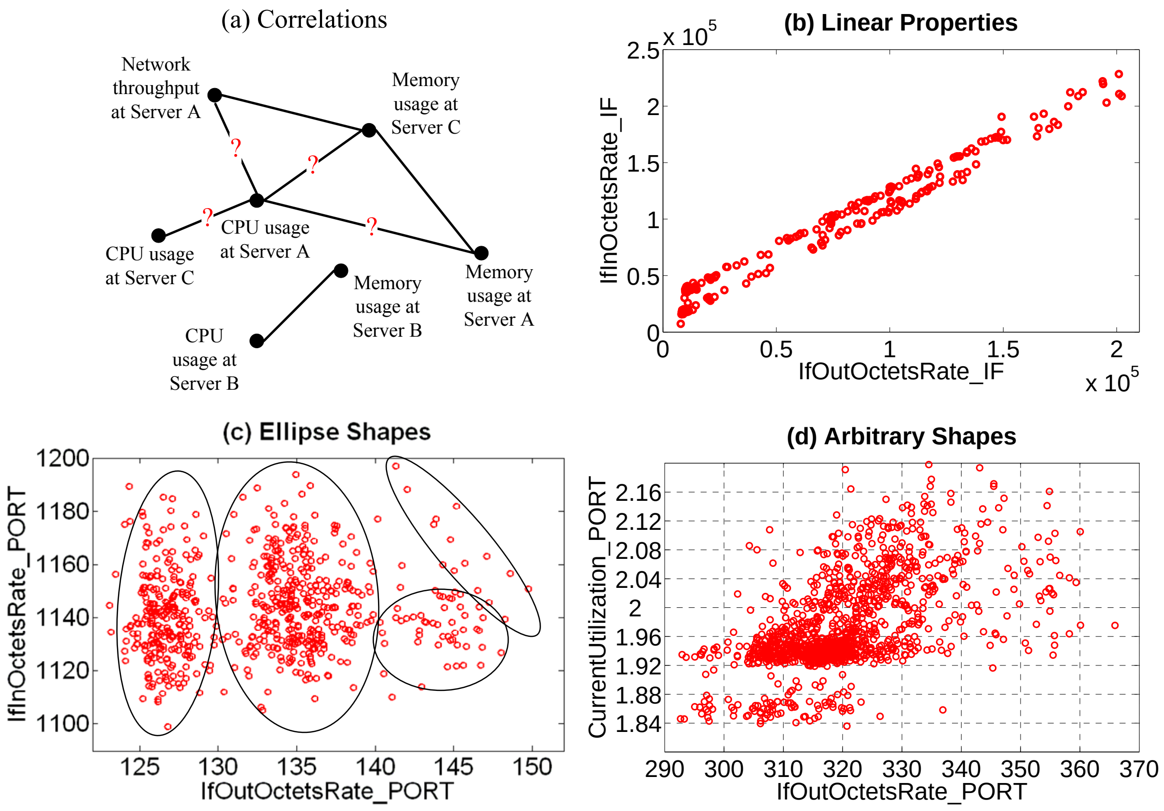


Figure 8.2: Pair-wise Measurement Correlations Shown in Two-dimensional Space

For the purpose of problem determination, it is essential to check the correlations among measurements instead of monitoring each measurement individually. A sudden increase in the values of a single measurement may not indicate a problem, as shown by the peaks in Figure 8.1(a) and Figure 8.1(b), instead, it could be caused by a flood of user requests. Monitoring multiple measurements simultaneously, we can identify this scenario as normal when we find that many measurements values increase but their correlations remain unchanged. Therefore, profiling measurement correlations can help find the “real” problems and reduce “false positives”. We are especially interested in tracking the pair-wise correlations, i.e., the correlations between any two measurements because it can assist quick problem localization. In Figure 8.2(a), we illustrate pair-wise correlations using a graph where each node represents a measurement and an edge indicates the correlation. At a certain time point, if all the links leading to a measurement m^a have certain problems, the system administrator can directly locate the problem source, i.e., m^a. The pair-wise correlations can be roughly divided into linear and non-linear categories. To observe the correlations more clearly, we extract the values of two measurements m^1 and m^2 at each time point t, and plot (m^1_t, m^2_t) as a point in the two-dimensional space. Figure 8.2(b)-(c) shows the measurement values extracted from real systems. Clearly, measurements in Figure 8.2(b) (the rate of traffic goes in and out the same machine), exhibit linear correlations, and Figure 8.2(c) (in and out traffic rate on two different machines), and Figure 8.2(d) (PORT throughput and utilization) demonstrate the non-linear relationships. In real monitoring data, we find that nearly half of the measurements have linear relationships with at least one of the other measurements, but the other half only have non-linear ones. Therefore, to model the behavior of the whole system, we need analysis tools that can identify both types of correlations.

Some efforts have been devoted to model the linear measurement correlations in distributed systems [94, 131]. Specifically, linear regression models are used to characterize the correlations, such as the one in Figure 8.2(b). Once the extracted linear relationship is broken, an alarm is flagged. In [79], the authors assume that the two-dimensional data points come from a Gaussian Mixture and use ellipses to model the data "clusters", so the points falling out of the cluster boundaries are considered anomalous events, as shown in Figure 8.2(c). Despite these efforts, there are many problems that restrict the use of the correlation profiling tools in real systems. First, existing work only focuses on one type of correlations, and thus cannot characterize the whole system precisely. Secondly, the assumption on the form of the data points may not be true (e.g., linear relationships or ellipse-shape clusters). For example, in Figure 8.2(d), the data points form arbitrary shapes and cannot be modeled by existing methods. Most importantly, how the data evolve is an important part of the system behavior, so besides spatial correlations, correlations across observation time should also be taken into consideration.

In light of these challenges, we propose a grid-based transition probability model to characterize correlations between any two measurements in a distributed system. As shown in Figure 8.2(d), we partition the space into a number of non-overlapping grid cells and map the data points into corresponding cells. A transition probability matrix is then defined over the two-dimensional grid structure where each entry V_{ij} corresponds to the probability of transitions from grid cell v_i to v_j. We initialize both the grid structure and the transition probability matrix from a snapshot of history monitoring data, e.g., collected from last month, and adapt them online to the distribution changes.

We then propose a fitness score to evaluate how well one or all the measurements are described by the correlation models. Once the fitness score drops below a threshold, it indicates that certain system problems may occur. Our contributions are: 1) We propose a novel probability model to characterize both spatial and temporal correlations among measurements from a distributed system. Based on the model, we develop methods to detect and locate system problems. 2) We make no assumptions on the type of correlations and data distributions, therefore, the proposed framework is general and can capture the normal behavior of the entire distributed system. Also, the model is easy to interpret and can assist later human debugging. 3) We demonstrate the proposed approach's ability of system problem detection and diagnosis by experimenting on one month's real monitoring data collected from three companies' IT infrastructures. We present the probability model in Section 8.2. Section 8.3 and Section 8.4 introduce how to compute and use the model. In Sections 8.5 and 8.6, we discuss experimental results and related work.

8.2 Transition Probability Model

At time t, the values of two system measurements m^1 and m^2 can be regarded as a two-dimensional feature vector $\mathbf{x}_t = (m_t^1, m_t^2)$. Then the task is to build a model M based on the incoming data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots$ to describe the correlations. Suppose $\mathbf{x}$ is drawn from $\mathcal{S} = \mathcal{A}^1 \times \mathcal{A}^2$, a 2-dimensional bounded numerical space. We partition the space $\mathcal{S}$ into a grid consisting of non-overlapping rectangular *cells*. We first partition each of the two dimensions into intervals. A *cell* is the intersection of intervals from the two dimensions, having a form $c = (v^1, v^2)$, where $v^a = [l^a, u^a)$ is one interval of $\mathcal{A}^a$ ($a \in \{1, 2\}$). A data point $\mathbf{x} = (m^1, m^2)$ is contained in the cell c if $l^a \leq m^a < u^a$ for $a = 1$ and $a = 2$. If $\mathcal{A}^1$ and $\mathcal{A}^2$ are partitioned into s^1 and s^2 intervals, there are altogether $s = s^1 \times s^2$ cells. The collection of all the non-overlapping rectangular cells is called *grid structure*: $\mathcal{G} = \{c_1, c_2, \ldots, c_s\}$.

We define the probability of having a new observation $\mathbf{x}_{t+1}$ based on $\mathcal{G}$. To simplify the problem, we assume that the future observation is only dependent on current value and not on any past ones (markov property), i.e., $P(\mathbf{x}_{t+1}|\mathbf{x}_t, \ldots, \mathbf{x}_1) = P(\mathbf{x}_{t+1}|\mathbf{x}_t)$. The experimental results in Section 8.5 show that this assumption works well in practice. Suppose $\mathbf{x}_{t+1} \in c_j$ and $\mathbf{x}_t \in c_i$, we then approximate $P(\mathbf{x}_{t+1}|\mathbf{x}_t)$ using $P(\mathbf{x}_{t+1} \in c_j|\mathbf{x}_t \in c_i)$, which is the probability of $\mathbf{x}_{t+1}$ falling into

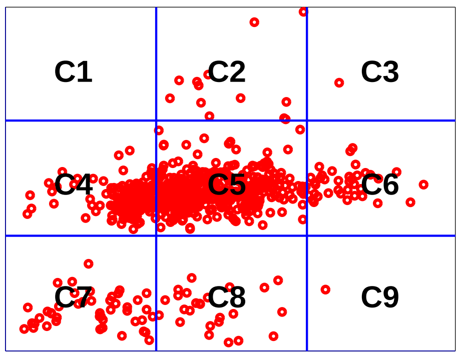


Figure 8.3: Grid Structure

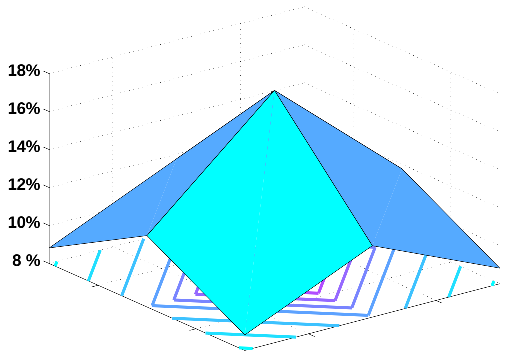


Figure 8.4: Transitions from c_5

Table 8.1: Transition Probability Matrix

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
c_1	21.98%	14.65%	8.79%	14.65%	10.99%	7.33%	8.79%	7.33%	5.49%
c_2	13.16%	19.74%	13.16%	9.87%	13.16%	9.87%	6.58%	7.89%	6.58%
c_3	8.79%	14.65%	21.98%	7.33%	10.99%	14.65%	5.49%	7.33%	8.79%
c_4	13.16%	9.87%	6.58%	19.74%	13.16%	7.89%	13.16%	9.87%	6.58%
c_5	8.82%	11.76%	8.82%	11.76%	17.65%	11.76%	8.82%	11.76%	8.82%
c_6	6.58%	9.87%	13.16%	7.89%	13.16%	19.74%	6.58%	9.87%	13.16%
c_7	8.79%	7.33%	5.49%	14.65%	10.99%	7.33%	21.98%	14.65%	8.79%
c_8	6.58%	7.89%	6.58%	9.87%	13.16%	9.87%	13.16%	19.74%	13.16%
c_9	5.49%	7.33%	8.79%	7.33%	10.99%	14.65%	8.79%	14.65%	21.98%

cell c_j when $\mathbf{x}_t$ belongs to cell c_i ($c_i, c_j \in \mathcal{G}$). To facilitate later discussions, we use $P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1})$ to denote $P(\mathbf{x}_{t+1}|\mathbf{x}_t)$, and use $P(c_i \rightarrow c_j)$ to denote $P(\mathbf{x}_{t+1} \in c_j|\mathbf{x}_t \in c_i)$. Since c_i and c_j are drawn from the collection of grid cells $\mathcal{G} = \{c_1, c_2, \ldots, c_s\}$, we can define a s by s matrix V where $V_{ij} = P(c_i \rightarrow c_j)$. Row i ($1 \leq i \leq s$) of the matrix V defines a discrete probability distribution $P(c_i \rightarrow c_j)$ ($\sum_{j=1}^{s} P(c_i \rightarrow c_j) = 1$) for the transitions from c_i to any cell in the grid ($c_j \in \mathcal{G}$). A snapshot of monitoring data from two measurements is plotted in Figure 8.3. The feature space is partitioned into nine grid cells: $c_1, c_2, \ldots, c_9$, and in Table 8.1, we show an example probability matrix $V_{9\times9}$. Suppose $\mathbf{x}_t$ is contained in cell c_5, the discrete probability distribution of $\mathbf{x}_{t+1}$ given $\mathbf{x}_t$ is then characterized by $V_{51}, V_{52}, \ldots, V_{59}$. As shown in Figure 8.4, higher probability on c_j indicates that $\mathbf{x}_{t+1}$ is more likely to jump to c_j when its original location is c_5.

Therefore, the model to characterize the pair-wise correlations consists of the grid structure and the probability matrix: $M = (\mathcal{G}, V)$. In Section 8.3, we discuss the methods to initialize and update the model. In Section 8.4, we describe how to use the model to determine system problems.

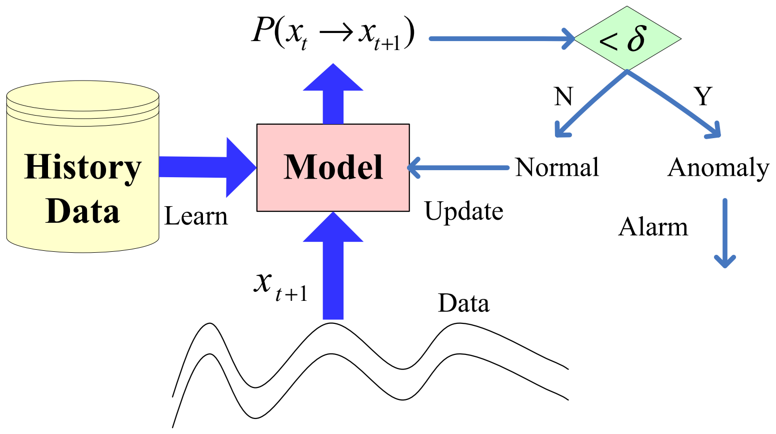


Figure 8.5: The Framework of Correlation Modeling

8.3 Model Computation

The framework of learning and updating the correlation probability model for problem determination is depicted graphically in Figure 8.5. We first initialize the model from a set of history data. The model is then put into use on the continuously flowing monitoring data. Based on the observed $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$, the model outputs $P(\mathbf{x}_t \rightarrow \mathbf{x}_{t+1})$ and if it is below a certain threshold δ, an alarm is flagged. We update the model to incorporate the actual transition made by $\mathbf{x}_{t+1}$ if it is normal. Since the model is comprised of grid structure $\mathcal{G}$ and probability matrix V, we present the learning algorithms for both of them as follows.

8.3.1 Grid Structure

Initialization. Based on a set of history data $\{\mathbf{x}_t\}_{t=1}^n$, we seek to design a grid structure $\mathcal{G}$, defined by a set of grid cells $\{c_1, c_2, \ldots, c_s\}$. Each cell is represented by a rectangle in the two-dimensional space. We compute the grid cells by setting their boundaries on the two dimensions separately. Formally, each cell c is defined as the intersection of any interval from each of the two dimensions, and the grid structure is thus represented by $\{(v_i^1, v_j^2)\}_{i=1,j=1}^{s^1,s^2}$, where v_i^1 and v_j^2 are intervals of A^1 and A^2 respectively.

Now the problem is: For data mapped onto one dimension a: $X^a = \{x_1^a, x_2^a, \ldots, x_n^a\}$, we wish to discretize $\mathcal{A}^a$ into s^a intervals to hold all the data points. We would compute transition probabilities based on the grid structure, so it should reflect the data distribution. Also, the computation needs to be efficient since multiple pairs of measurements may be watched. Therefore, we propose an efficient approach to partition each dimension into intervals adaptive to the data distribution based on MAFIA [77], a clustering method. We first get the upper and lower bound l^a and u^a from X^a and divide $[l^a, u^a)$ into small equal-sized units with unit length z^a. Note that z^a is much smaller than the actual interval size of the grid structure. We count the number of points falling into each unit. Adjacent units are then merged to form an interval if their counts are similar with respect to a threshold, or are both below a density threshold. The basic idea behind this is to represent the dense areas using more cells, and regions with similar probability densities can be represented using one cell because they may have similar transition patterns. If the data are equal-distributed, we ignore the above procedure and simply divide the dimension into equal-sized intervals. We run the above procedure for each dimension and obtain all the cells by intersecting intervals of the two dimensions. Figure 8.6 shows an example of the history data and the grid structure the algorithm generates for the data.

Update. During the online process, most of the time, a new observation $\mathbf{x}_{t+1}$ falls into one of the cells defined by the grid structure $\mathcal{G}$. However, it is likely that $\mathbf{x}_{t+1}$ is out of the boundary defined by $\mathcal{G}$. Then either $\mathbf{x}_{t+1}$ is an outlier, or the underlying distribution has changed. We wish to ignore the outliers, but only adapt the grid structure according to the distribution evolution. However, it is challenging to distinguish between the two cases in a real-time manner. We observe that real data usually evolve gradually, thus we assume that the boundary of the grid structure is also changing gradually. Therefore, when $\mathbf{x}_{t+1}$ is not contained in any cells of $\mathcal{G}$, we only update $\mathcal{G}$ if $\mathbf{x}_{t+1}$ is *close enough* to the grid boundary. For each dimension A^a, we compute the average interval size r^a_{avg} offline during initialization and suppose the upper bound of $\mathcal{G}$ on dimension A^a is u^a. When $x^a_{t+1} > u^a$ for $a = 1$ or 2, we first judge if $x^a_{t+1} \leq u^a + \lambda^a \cdot r^a_{avg}$, where λ^a is a parameter indicating the maximum number of intervals to be added. If it holds true, we take it as a signal of potential distribution evolution and add intervals to the dimension until $\mathbf{x}_{t+1}$ is contained within the boundary. New cells are incorporated into $\mathcal{G}$ as the intersections of the added intervals and the intervals from the other dimension. Note that we do not delete cells having sparse densities to maintain the rectangular shape of the grid structure for fast computation. Figure 8.7 shows the online data and the accordingly updated grid structure, whereas the offline structure is illustrated

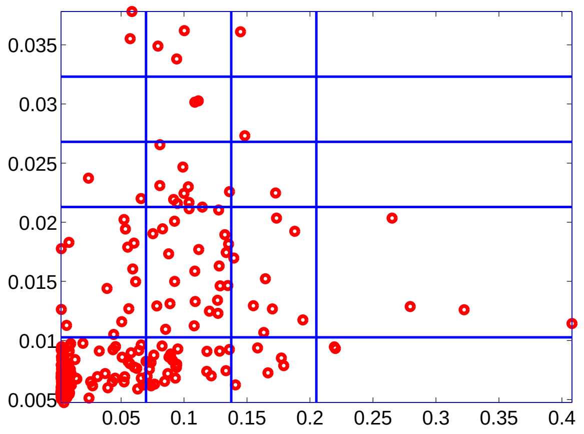


Figure 8.6: Initial Grid

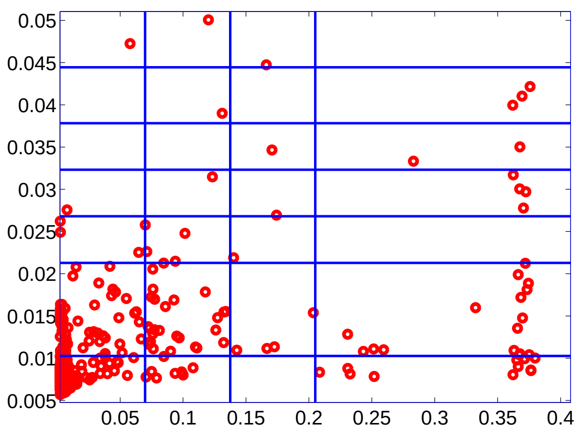


Figure 8.7: Updated Grid

in Figure 8.6. It can be seen that the data evolve along the vertical axis, and thus two more intervals are added to accommodate such changes.

8.3.2 Transition Probability Matrix

We seek to compute $P(c_i \to c_j)$ for any c_i and c_j in $\mathcal{G}$, i.e., the transition probability between any pair of cells. One natural solution is to compute the empirical distribution based on the set of monitoring data D. Specifically, let $P(c_i \to c_j)$ be the percentage of examples jumping to c_j when it originally stays at c_i. Although the empirical probability can capture most transitions, it may not be accurate on the transitions which are under represented or even unseen in past records. We therefore need to adjust the empirical distribution to make it smooth over the space so that an unseen transition may still have chances to occur in the future. Therefore we introduce a prior into the distribution using the following bayesian analysis technique [71]: $P(c_i \to c_j|D) = \frac{P(D|c_i \to c_j)P(c_i \to c_j)}{P(D)}$ where $c_i \to c_j$ indicates the existence of a transition from cell c_i to c_j and D is monitoring data set. The transitions are assumed to be independent of each other. Also, our aim is to infer $c_i \to c_j$, so the term $P(D)$ is not relevant and can be omitted:

$$P(c_i \to c_j|D) \propto P(c_i \to c_j) \prod_{t=1}^{n-1} P(\mathbf{x}_t \to \mathbf{x}_{t+1}|c_i \to c_j) \tag{8.1}$$

where n is the size of D. The two steps under this bayesian framework include: 1) define a prior distribution for $P(c_i \to c_j)$ for any c_i and c_j, and 2) update the distribution based on each observed transition from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. After all data points in D are seen, we can obtain the posterior probability $P(c_i \to c_j|D)$. We explain the two steps as follows.

Prior Distribution. Bayesian methods view the transition from c_i to c_j as a random variable having a prior distribution. Observation of the monitoring data converts this to a posterior distribution. When a transition is seldom or never seen in the data, the prior will play an important role. Therefore, the prior should reflect our knowledge of the possible transitions. The question is, given $\mathbf{x}_t \in c_i$, which cell is the most probable of containing $\mathbf{x}_{t+1}$? With respect to our assumption that the monitoring data evolve gradually, the transition would have the "*spatial closeness tendency*", i.e., the transitions between nearby cells are more probable than those between cells far away. To support this claim, we check the number of transitions with respect to the cell distance in two days' measurement values. We find that the total number of transitions is 701, among which 412 occurs inside the cells, i.e., the data points would simply stay inside a certain cell. There are 280 transitions between a cell and its closest neighbor. As the cell distance increases, it becomes less likely that points move among these cells. Therefore, the "spatial closeness tendency" assumption is valid. Based on this finding, we define the prior distribution as $P(c_i \to c_j) \propto \frac{P(c_i \to c_i)}{w^{d(c_i,c_j)}}$ where $d(c_i, c_j)$ is the distance between c_i and c_j, and w is the rate of probability decrease. If we observe that $\mathbf{x}_t$ belongs to c_i, it is most likely that $\mathbf{x}_{t+1}$ stays at c_i as well. We set $P(c_i \to c_i)$ to be the highest and as c_j departs further away from c_i, $P(c_i \to c_j)$ decreases exponentially. From the definition and the constraints that $\sum_{j=1}^{s} P(c_i \to c_j) = 1$, the prior probability of having transitions from c_i to any cell can be computed. An example prior distribution of transiting from cell c_{12} to other cells is shown in Figure 8.8. It can be seen that the transition probability at c_{12} is the highest, followed by the probability of transitions to its closest neighbors.

Distribution Updates. According to Eq. (8.1), to update the prior distribution, we need to multiply it by $P(\mathbf{x}_t \to \mathbf{x}_{t+1}|c_i \to c_j)$. If x_{t+1} in fact falls into c_h, we should set $P(x_t \to x_{t+1}|c_i \to c_h)$ to be the highest among all the pairs of cells. Also, due to the "spatial closeness tendency", it is likely that a future transition can occur from c_i to c_h's neighbors. Again, we assume an exponential decrease in the transition probability with respect to the cell distance and use the following update

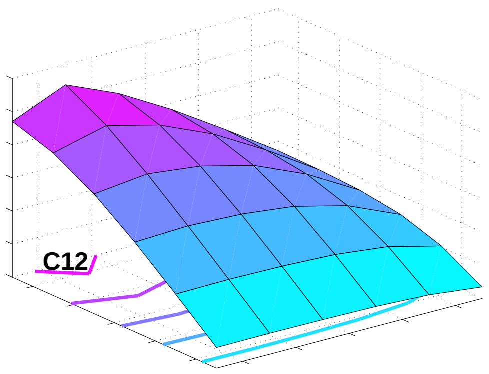


Figure 8.8: Initial Transitions

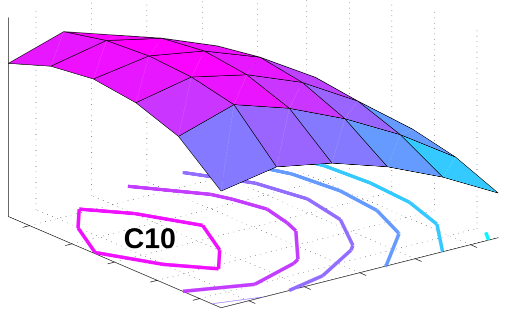


Figure 8.9: Updated Transitions

rule:

$$P(\mathbf{x}_t \to \mathbf{x}_{t+1}|c_i \to c_j) \propto \frac{P(\mathbf{x}_t \to \mathbf{x}_{t+1}|c_i \to c_h)}{w^{d(c_h,c_j)}}$$
$$\text{if } \mathbf{x}_{t+1} \in c_h \text{ and } \mathbf{x}_t \in c_i \qquad (8.2)$$

On Eq. (8.1), we take log over all the probabilities, and the updates can be performed using additive operations. Note that we update the transition probability only on normal points, but not on outliers with zero probability. The updating equation is applied on the i-th row of the transition probability matrix where c_i is the cell $\mathbf{x}_t$ belongs to. We start the updating procedure from x_1 where $P(c_i \to c_j|x_1)$ is assumed to be the prior: $P(c_i \to c_j)$, and repeatedly execute it for $i = 2, \ldots, n-1$. The prior distribution shown in Figure 8.8 is updated using six days' monitoring data and the posterior probability distribution on cell c_{12} is depicted in Figure 8.9. The prior probability of going from c_{12} to c_{12} is the highest, but it turns out that many transitions from c_{12} to c_{10} are observed, so the probability at c_{10} becomes the highest in the posterior.

8.4 Problem Determination and Localization

In this section, we discuss how to determine problems in a distributed system with l measurements available. Since we build pair-wise correlation models for any two measurements, we have $l(l-1)/2$ models to characterize all the correlations within the whole system. We propose a *fitness score* as

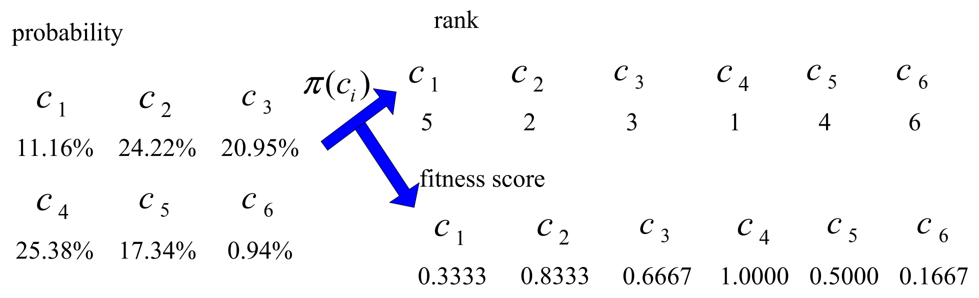


Figure 8.10: Fitness Score Computation

an indicator for the probability of having system problems, which is defined at the following three levels and measures how well the models fit the monitoring data.

1) ***Each pair of measurements at a given time***: For a pair of measurements m^a and m^b at time $t+1$, suppose the most updated model derived from the monitoring data from time 1 to t is $M_{t+1}^{a,b}$. $\mathbf{x}$ represents the two dimensional feature vector consisting of measurement values from m^a and m^b. Suppose $\mathbf{x}_t$ falls into cell c_i. At time $t+1$, the model $M_{t+1}^{a,b}$ outputs the transition probability from c_i to any cell c_j in the grid ($1 \leq j \leq s$). We define a ranking function $\pi(c_j) : \pi(c_j) < \pi(c_k)$ if $P(c_i \rightarrow c_j) > P(c_i \rightarrow c_k)$. In other words, c_j would be ranked higher if the probability of going from c_i to c_j is higher. We then define the fitness score as $Q_{t+1}^{a,b} = 1 - \frac{\pi_{M_{t+1}^{a,b}}(c_h)-1}{s_{M_{t+1}^{a,b}}}$ where c_h is the cell $\mathbf{x}_{t+1}$ actually belongs to, and $s_{M_{t+1}^{a,b}}$ is the number of grid cells in model $M_{t+1}^{a,b}$. Outliers that lie outside the grid have zero transition probability, and thus their fitness scores are zero as well. Figure 8.10 illustrates the fitness score computation through an example. Suppose $\mathbf{x}_t$ is contained in cell c_4 and the transition probability from c_4 to other cells is shown in the left part of the figure. If $\mathbf{x}_{t+1}$ is in cell c_5, we first sort the cells according to the transition probability and c_5 is ranked at the 4-th place. Then to compute the fitness score, we have $\pi_{M_{t+1}^{a,b}} = 4$ and $s_{M_{t+1}^{a,b}} = 6$, so the result is 0.5. To examine the effect of fitness scores, we repeat the above procedure for the other cells and the results are shown in Figure 8.10. As can be seen, the fitness score Q measures the fitness of model $M_{t+1}^{a,b}$ on the observed monitoring data. 2) ***Each measurement at a given time***: For a measurement $m^a (1 \leq a \leq l)$, we can derive $l-1$ different models, each of which characterizes the correlations between m^a and m^b ($b = 1, \ldots, a-1, a+1, \ldots, l$). At time $t+1$, the fitness score for m^a is computed as: $Q_{t+1}^a = \frac{\sum_{b \neq a} Q_{t+1}^{a,b}}{l-1}$ where $Q_{t+1}^{a,b}$ is the fitness score for the model built upon m^a and another measurement m^b. The fitness score of a single measurement is determined by the

fitness of correlation models constructed for its links to all the other measurements. 3) ***At a given time***: We aggregate the scores from l measurements into one score Q_{t+1}, which can be used to judge if there are any problems in the entire system at time $t+1$. Again, this can be achieved by averaging the fitness scores of all the measurements.

At the finest level, $Q_{t+1}^{a,b}$ only evaluates the correlation model between two measurements (e.g., one link in Figure 8.2(a), such as the link between "CPU Usage at Server A" and "Memory Usage at Server C"). Q_{t+1}^{a} is the aggregation of $Q_{t+1}^{a,b}$, i.e., examining the $l-1$ links leading to one node. For example, the fitness score for measurement "CPU Usage at Server A" is computed based on all its links. Q_{t+1} works for the entire system by aggregating all the fitness scores (e.g., all the links in Figure 8.2(a)). In general, this evaluation framework can provide different granularity in the data analysis for system management. We can first merge the fitness scores of all the system components so that the system administrators can monitor a single score for system-wide problems. If the average score deviates from the normal state, the administrators can drill down to Q_{t+1}^{a} or even $Q_{t+1}^{a,b}$ to locate the specific components where system errors occur. We can expect a high fitness score when the monitoring data can be well explained by the model, whereas anomalies in system performance lead to a low score.

8.5 Experiments

We demonstrate the effectiveness and efficiency of the proposed method through experiments on a large collection of real monitoring data from three companies' infrasturcture. Due to privacy issue, we cannot reveal their names and will denote them as A, B and C in the following discussions. Each company provides a certain Internet service and has over a hundred servers to support user requests every day. On each server, a wide range of system metrics are monitored that are of interests to system administrators, for example, free memory amount, CPU utilization, I/O throughput, etc. A metric obtained from a machine represents a unique measurement. For example, CPU utilization on machine with IP "x.x.x.x" is one measurement. We expect that correlations exist among measurements from the same machine, as well as across different machines, because the whole system is usually affected by the number of user requests.

For each group, there are roughly 3000 measurements collected from around 150 machines. We

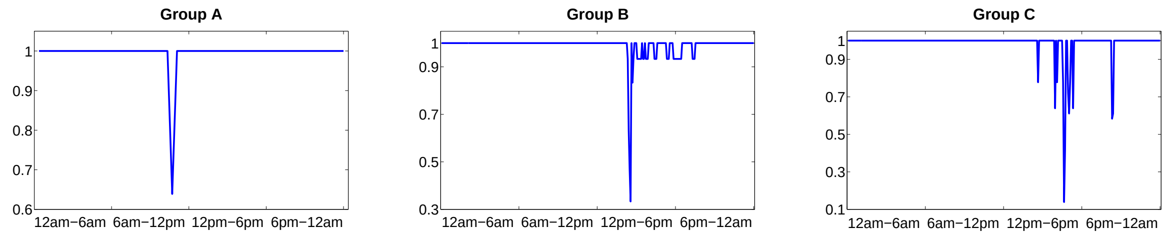


Figure 8.11: Fitness Scores When System Problems occur

select 100 from each group and conduct the experiments on the $3 \times \binom{100}{2}$ pairs of measurements. To test on the difficult cases, we enforce the following selection criteria: 1) The sampling rate should be reasonably high, at least every 6 minutes; 2) The measurements do not have any linear relationships with other measurements; and 3) The measurement should have high variance during the monitoring period. We wish to find out the proposed transition probability model's ability in profiling the system normal behavior. To achieve this, we sample a training set to simulate history data, and a test set, which can be regarded as online data, from the one month's monitoring data (May 29 to June 27, 2008). We compute a model from the training set and evaluate it on the test set. To examine how the sizes of the training and test set affect the model performance, we construct the following training and test sets and conduct experiments on all the combinations for each of the three groups. ***Training sets***: 1) 1 day (May 29), 2) 8 days (May 29-June 5), and 3) 15 days (May 29-June 12). ***Test sets***: 1) 1 day (June 13), 2) 5 days (June 13-June 17), 3) 9 days (June 13-June 21), and 4) 13 days (June 13-June 25).

Problem Determination. In this part, we assess the performance of the proposed method in system problem determination. The distributed systems in use are usually stable and do not have any critical failures. Therefore, we test our methods on three pairs of system measurements where potential problems occur as identified by the system administrators. Based on these events, we can get some general ideas about the proposed method's effectiveness in problem determination. Figure 8.11 depicts the fitness scores for three pairs of measurements where the ground-truth problems are found. The test set is one day's monitoring data and the problems are found in the morning (Group A), or in the afternoon (Group B and C). The two measurements are CurrentUtilization_PORT and ifOutOctetsRate_PORT (Group A), ifOutOctetsRate_PORT and ifInOctetsRate_PORT (Group B),

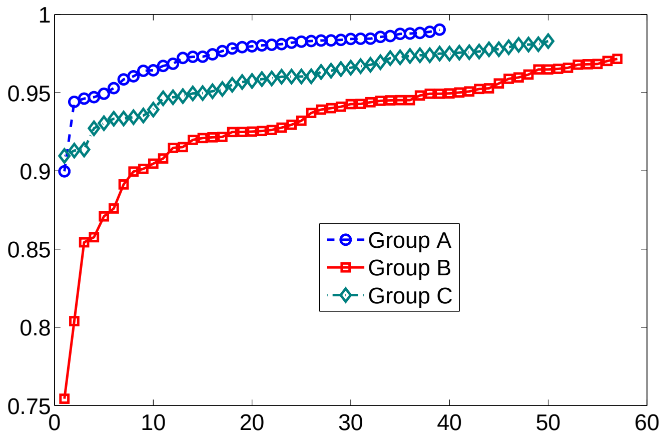


Figure 8.12: Q Scores w.r.t Locations

and CurrentUtilization_IF and ifOutOctetsRate_IF (Group C). It clearly shows that the anomalies identified by the proposed transition probability method are consistent with the ground-truth in all the three cases. During the period when a problem occurs, we can observe a deep downward spike in the plot of fitness score, which means that this problematic time stamp receives a much lower fitness score compared with normal periods. To provide some intuitive ideas about how the method detects these anomalies, we show the normal and anomalous transitions for the experiments on Group B. From 12am up to 2pm, the values of the two measurements stay within the normal ranges [47.321,22588] & [88.83,34372], however, an anomalous jump to the grid cell [22588,45128] & [102940,137220] is observed, which leads to the downward spike in the fitness score. After that, the measurements fall into either the above normal ranges or [22588,67670] & [34372,51510], which gives a little disturbance to the fitness scores until 8pm. Finally, the measurements go back to their normal values and thus the fitness score stabilizes at 1. Note that we omit the transition probability here, but only give the normal and anomalous transitions to illustrate the basic idea. So the proposed model can help detect the system problems as well as investigate the problem causes.

We also try to identify the specific machine where the problem locates within the whole distributed system. To do so, we compute the average fitness score among measurements collected from the same machine and plot the score distribution across each information system in Figure 8.12. The locations with low fitness scores are the potential problem sources. Because the mon-

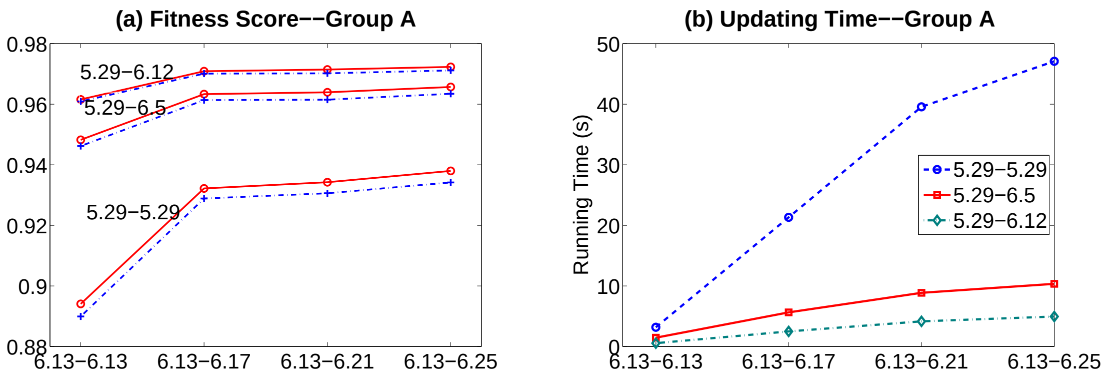


Figure 8.13: Average Fitness Score and Updating Time

itoring data from the three information systems have different characteristics and distributions, the scales of fitness scores on the three groups are different. We can see that most of the fitness scores are above a certain threshold within each group, which implies that most of the servers are stable and have few problems. There are only a few servers with low average scores, where the system administrators need to check carefully. For example, in Figure 8.12, there is only 1 machine scoring at below 0.9 in group A, much lower than the scores of other machines. We should pay more attention to this server in future monitoring and analysis.

Offline versus Adaptive. In the following experiments, we show the method's performance on all pairs of measurements by analyzing the fitness scores. As discussed, the real distributed system exhibits normal behavior most of the time, therefore, a good model should predict the system behavior well and generate a high average fitness score. First, we compare the following two methods: ***Offline*** methods where the model is derived from the training set offline, and ***Adaptive*** methods where the model is initialized from the training set but updated based on online test set. It would be interesting to see if online model updating can provide additional benefits to the offline model. In Section 8.4, we show that, at each sampling point, a fitness score Q_{t+1} is computed to reflect the effectiveness of the current model. Therefore, we can evaluate the performance of offline and adaptive methods by averaging the fitness scores computed according to their generated model at each time stamp. When the model is continuously good, the average fitness score would be high. Due to the space limit, we only show the experimental results on group A. The experiments on the other two groups have similar patterns. The results are shown in Figure 8.13(a), where solid and

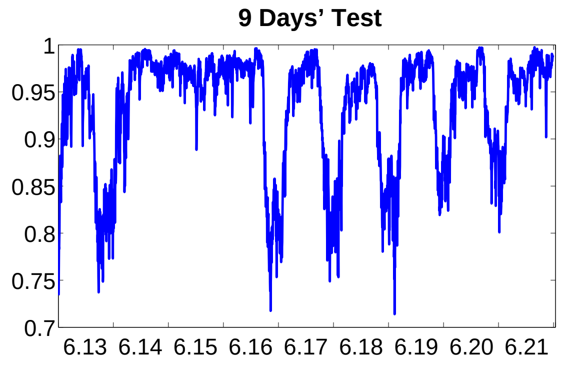


Figure 8.14: Q Scores for Nine Days

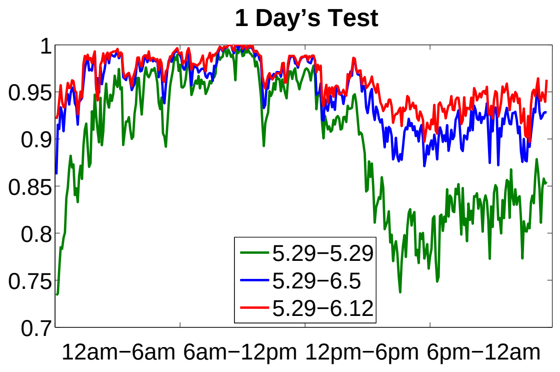


Figure 8.15: Q Scores for One Day

dotted lines represent adaptive and offline methods respectively. It can be seen that the adaptive method usually improves the fitness score over the offline method, especially when the training set is small. When history data are limited, online updating of the model is necessary. But when we have sufficient history data, the models from offline analysis can predict reasonably well on the test set. When the size of the test set increases, we can observe an increase in the fitness scores, which can be explained by the fact that large sample size usually reduces the estimates' variance. Typically, the average fitness score is between 0.8 and 0.98, indicating that the proposed model captures the transitions in monitoring data and is capable of predicting the future.

Updating Time. In this part, we evaluate the adaptive method's efficiency. First, once we have a new observation, we simply determine the grid cell it falls into and look up the transition probability matrix to get the prediction, so the time of applying the model to make predictions is negligible. On the other hand, we have relatively more time to spend for offline analysis. Therefore, the time of updating the model online is the most important part in efficiency analysis. Figure 8.13(b) shows the online updating time of the adaptive method. When the training samples are sufficient (9 days or 15 days), it costs below 10 seconds to process more than 4,000 monitoring data points, i.e., less than 2.5 milliseconds per sample, much smaller than the sampling frequency (6 minutes). If the period of the training set drops to one day, the updating time increases greatly. Because the history data set does not contain enough examples to initialize the model accurately, the model has to be updated frequently online. However, even in the worst case, the updating time is less than 23 milliseconds per sample. So the proposed method is efficient and can be embedded in online monitoring tools.

Periodic Patterns. The volume of user requests usually affects the system behavior. Heavier work loads can make the system less predictable. Therefore, when we examine the fitness score at each time stamp Q_{t+1} over a period of 9 days, we find some interesting periodic patterns in Figure 8.14. We initialize the model using one day's monitoring data, then update and evaluate it on the data from June 13 to June 21. It is obvious that higher fitness scores are obtained during the time when the system is less active including the weekends. At peak hours, the model has lower fitness scores because the system is heavily affected by the large volume of user requests and would be difficult to predict correctly. When more history data are employed in building the initial model, the fitness scores can be improved greatly. To illustrate this, we vary the size of the training set and plot the fitness scores on one day's monitoring data (June 13), shown in Figure 8.15. When only one day's data are used as training set, the fitness score drops when heavy workloads increase the prediction complexity. But the model initialized from 15 days' history data greatly improves the stability, with a fitness score above 0.9 during both peak and non-peak hours. The results suggest that it is important to incorporate more training samples that share similar properties with the online data to learn the initial model.

8.6 Related Work

Due to the increase in complexity and scale of the current systems, it becomes important to utilize the measurement correlation information in system logs for autonomic system management. Methods are developed to model correlations of request failures [35], or among server response time [7]. Correlating monitoring data across complex systems has been studied recently, when algorithms are developed to extract system performance invariants [94, 131] and describe the non-linear correlations [79]. Our proposed method distinguishes itself from the above methods by modeling both spatial and temporal correlations among measurements. In markov model based failure prediction methods [143], the temporal information is taken into consideration, but they require the event-driven sources, such as system errors as input. Conversely, our method does not require any knowledge about the system states. The problem of anomaly detection has been extensively studied in several research fields. Particularly, many algorithms have been developed to identify faults or intrusions in Internet [117] or wireless network [36] by examining network traffic data. Different from the above methods, our approach models the data evolution instead of static data points, and thus detects outliers from both spatial and temporal perspectives. In the proposed framework, we partition the two-dimensional data space into grid cells. The idea of space partitioning is motivated by grid-based clustering algorithms [77]. The term “grid” refers to the resulting discretized space, thus carries a completely different meaning from that in “grid computing”.

8.7 Summary

In this chapter, we describe a novel statistical approach to characterize the pair-wise interactions among different components in distributed systems. We discretize the feature space of monitoring data into grid cells and compute the transition probabilities among the cells adaptively according to the monitoring data. Compared with previous system monitoring techniques, the advantages of our approach include: 1) It detects the system problems considering both spatial and temporal information; 2) The model can output the problematic measurement ranges, which are useful for human debugging; and 3) The method is fast and can describe both linear and non-linear correlations. Experiments on monitoring data collected from three real distributed systems involving 100 measurements from around 50 machines, show the effectiveness of the proposed method.

Chapter 9

Conclusions and Future Work

We now live in a connected world, and the ability to collect and make effective use of multiple information sources in different formats will pave the way for success in the coming decades. Mining multiple information sources *simultaneously* is the key solution to effective knowledge acquisition from the gigantic data collections we have. It has two major benefits: 1) Although the amount of data continues to grow at an astounding rate, most of them contain erroneous, corrupted, or missing entries due to many reasons, such as unreliable data acquisition sources, faulty sensors, and data collection errors. Combining different channels of information can average out independent errors in each source, give a global picture of the mined knowledge and thus provide a robust solution to mining low-quality data; 2) There exists some hidden knowledge that can only be found across information sources, and thus we must connect different pieces of information in various format and their different solutions, and pay particular attention to their interactions.

In this thesis, we presented a series of algorithms that take advantage of rich information contained in multiple information sources to help with the task of classification and anomaly detection in challenging situations. By synthesizing or comparing multiple information channels, we can identify insightful knowledge and provide users a robust and accurate solution. In this chapter, we come to the conclusions of this thesis and discuss the future directions of mining multiple information sources.

9.1 Summary

In this thesis, we presented our solutions for the joint discovery of useful knowledge from multiple information sources. In general, we made the following contributions.

- **Systematic Study of Learning from Multiple Sources**

We proposed to utilize complementary information from multiple sources for better knowledge discovery and proposed two general learning frameworks. The proposed frameworks can integrate the available knowledge obtained from all the information sources (Chapter 3), and identify meaningful and unexpected anomalies from heterogeneous information sources by exploring inconsistencies (Chapter 6). These two general learning frameworks greatly advance the studies of learning from multiple sources and can benefit a variety of real applications.

- **Benefits of Multiple Source Mining**

  We demonstrated the benefits of analyzing multiple information sources simultaneously from both theoretical and experimental perspectives. The advantages of multi-source mining were shown in a variety of difficult learning scenarios including transfer learning (Chapter 4), dynamic stream mining (Chapter 5), information networks (Chapter 7) and system debugging (Chapter 8). We systematically studied and analyzed multi-source mining in both knowledge synthesization and inconsistency detection, and gave robust solutions to these challenging problems.

- **Algorithms**

  As multi-source mining provides great benefits to knowledge discovery, we developed several core algorithms for integrating knowledge from multiple data sources. Our major contributions include a consensus maximization method for multiple source integration (Chapter 3) and a spectral framework to detect inconsistency across multiple sources (Chapter 6). Based on the same principle to analyze multiple sources simultaneously, we developed a locally weighted ensemble framework for transfer learning (Chapter 4), a model averaging approach for stream data classification (Chapter 5), probabilistic modeling of community outliers (Chapter 7) and a statistical model of measurement correlations in distributed systems (Chapter 8). These algorithms provide highly efficient and effective solutions for the several most important data mining problems in multiple information source environment.

- **Applications**

  The proposed methods combine heterogeneous channels of information and thus provide robust and accurate solutions. We evaluate each proposed approach on both synthetic and real-world data, and the experiments show that each approach achieves high accuracy, efficiency, and scalability. The real data sets cover a variety of applications including text mining (Chapters 3, 4, 6, 7), security and networking (Chapters 3, 5, 8), sentiment analysis (Chapter 3), computer science bibliography (Chapters 3, 6, 7), and social media (Chapters 3, 6). Besides these applications, the algorithms have the potential of being applied to many different fields where multiple diverse information sources are available to capture object properties and similarity relationships, for example, healthcare, bioinformatics, business intelligence and energy efficiency.

9.2 Future Directions

We envision that the infrastructure and technologies for active storage, extraction, exchange and modeling among multiple heterogeneous information sources will be the center of interests in the future. In the past decade, many efforts have been devoted to developing methods to obtain knowledge from multiple information sources, but as new challenges emerge, effective multi-source mining approaches are in great demand. In this thesis, we presented approaches that address several challenges including lack of supervision, dynamically evolving data and privacy and security concerns. To further advance the field, there are several important research issues that should be explored.

- **Lack of Uniform Structured Representation**

  Data pre-processing is an often neglected but important step in the data mining process. In heterogeneous multiple sources, data can have inconsistent representations, their values can be corrupted and noisy, and some of them even don't have structured representations. There are numerous applications that lack proper data representations, including chemical compounds, proteins, image data, social networks and transaction data. A possible solution is to extract relevant and frequent patterns from multiple sources as structured, consistent and robust representations. The basic principle is to extract a compact set of frequent patterns that are relevant to the task, but meanwhile, the extracted patterns should represent the intrinsic property of each individual source as well as the interactions between sources as much as possible.

- **Existence of Noisy Information Sources**

  In our work, we take the relative importance of each data source into consideration, however, we still assume that each source is at least relevant to the data mining task. In reality, there may exist multiple noisy, heterogeneous and incomplete information sources and some of them may be irrelevant to the task. Incorporating such sources into the combination framework may degrade the performance, and thus a careful filtering procedure should be performed. Emerging applications bring in new challenges to select the right sources due to the complicated interactions among sources. This problem shares some basic principles with feature selection, where a set of representative features are chosen to improve classification accuracy. The selected features should be relevant to the classification task, and they should be uncorrelated to provide complimentary knowledge about the task. To generalize the idea of feature selection to source selection, we need to develop statistical measures to compare the data distributions, quantify the trustworthiness, as well as select the right subset of information sources.

- **Large-Scale Data**

  Scalability is another important issue in mining multiple sources since each information source may possess giga, tera or even peta-bytes of data. It might be natural to apply parallel computing framework to solve multi-source mining problems, but some data mining tasks may require interactions among sources and thus we have to design parallel computing methods that minimize communication costs. In general, there are two ways to conquer the problem: Separate the data into multiple subsets and distribute them across machines, or compress the data along certain dimensions or sample a subset of data to reduce the size. While the first approach ignores the interactions among sources, important details might be lost during compression in the second approach. Therefore, it is desirable to leverage the two approaches by distributing and compressing data to a certain extent while communicating minimal amount of necessary information across machines. To minimize the information loss and communication cost, we need to carefully select the dimensions along which we compress data, as well as the data sampling and partition strategy. The decisions would vary according to the data characteristics and the nature of the task. It is worth investigating the trade-off between accuracy and efficiency by developing a cost model, and selecting the appropriate strategy to parallelize the algorithm according to the model.

- **Difficulty in Understanding Mining Results**

  As data mining is used to solve problems in real practice, it is extremely important to have the results interpretable and accessible to users with limited background knowledge in data mining. Users want to know why and where the solutions come from. Different data mining tasks pursue different goals of post-processing: Interpreting frequent patterns requires a succinct and meaningful set of summarized patterns, whereas it is ideal to locate the path that leads to anomalies in anomaly detection. Interpreting results from multiple sources is more challenging, but can provide a more insightful and vivid knowledge representation. Instead of learning from raw data, we try to synthesize the obtained knowledge. A variety of techniques, such as rule mining, clustering, user-guided exploration and other statistical summarization methods should be combined together to better illustrate and visualize the results, locate the root causes of the problems as well as provide semantically enriched and in-depth descriptions of the knowledge obtained from heterogeneous information sources.

It will be exciting to build a next-generation data management and analysis system, which transforms multiple sources of gigantic, noisy, heterogeneous, complicated data into accurate, reliable, and accessible knowledge. To deliver such a complete solution, there are numerous opportunities in mining and management of multiple information sources.

References

[1] N. Abe. Sampling approaches to learning from imbalanced datasets: active learning, cost sensitive learning and beyond. In *Proc. of the ICML-KDD'03 Workshop: Learning from Imbalanced Data Sets*, 2003.

[2] C. Aggarwal, J. Han, J. Wang, and P. S. Yu. On demand classification of data streams. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 503–508, 2004.

[3] A. Argyriou, C. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems (NIPS'08)*, pages 25–32, 2008.

[4] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 164–169, 1996.

[5] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[6] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS'02)*, pages 1–16, 2002.

[7] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. *ACM SIGCOMM Computer Communication Review*, 37(4):13–24, 2007.

[8] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[9] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 59–68, 2004.

[10] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. *ACM Computing Surveys*, 41:16:1–16:52, 2009.

[11] G. Batista, R. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter*, 6:20–29, 2004.

[12] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 2004.

[13] S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery (DMKD)*, 5:213–246, 2001.

[14] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS'07)*, pages 137–144. 2007.

[15] P. N. Bennett, S. T. Dumais, and E. Horvitz. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8(1):67–100, 2005.

[16] D. P. Bertsekas. *Non-Linear Programming.* Athena Scientific, 2 edition, 1999.

[17] J. Besag. Spatial interaction and the statistical analysis of lattic systems. *Journal of the Royal Statistical Society, Series B*, 36(2):192–236, 1974.

[18] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48(3):259–302, 1986.

[19] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1, 2007.

[20] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proc. of the International Conference on Machine Learning (ICML'07)*, pages 81–88, 2007.

[21] S. Bickel and T. Scheffer. Multi-view clustering. In *Proc. of the IEEE International Conference on Data Mining (ICDM'04)*, pages 19–26, 2004.

[22] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[23] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of the Annual Conference on Computational Learning Theory (COLT'98)*, pages 92–100, 1998.

[24] N. Borlin. Implementation of hungarian method. http://www.cs.umu.se/∼niclas/matlab/assignprob/.

[25] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[26] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[27] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pages 93–104, 2000.

[28] A. J. Carlson, C. M. Cumby, J. L. R. Nicholas D. Rizzolo, and D. Roth. Snow learning architecture. http://l2r.cs.uiuc.edu/˜cogcomp/asoftware.php?skey =SNOW#projects.

[29] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[30] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proc. of the International Conference on Machine Learning (ICML'04)*, pages 137–144, 2004.

[31] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41:15:1–15:58, 2009.

[32] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[33] K. Chaudhuri, S. M. Kakade, K. Livescu, and K. Sridharan. Multi-view clustering via canonical correlation analysis. In *Proc. of the International Conference on Machine Learning (ICML'09)*, pages 129–136, 2009.

[34] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Newsletter*, 6:1–6, 2004.

[35] M. Chen, E. Kiciman, E. Fratkin., A. Fox, and E. Brewer. Pinpoint: problem determination in large, dynamic internet services. In *Proc. of the International Conference on Dependable Systems and Networks (DSN'02)*, pages 595–604, 2002.

[36] P. Chhabra, C. Scott, E. Kolaczyk, and M. Crovella. Distributed spatial anomaly detection. In *Proc. of the IEEE International Conference on Computer Communications (INFOCOM'08)*, pages 1705–1713, 2008.

[37] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 2007.

[38] G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *Proc. of the SIAM International Conference on Data Mining (SDM'05)*, pages 44–54, 2005.

[39] F. Cozman and I. Cohen. Semi-supervised learning of mixture models. In *Proc. of the International Conference on Machine Learning (ICML'03)*, pages 99–106, 2003.

[40] K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9:1757–1774, 2008.

[41] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu. Co-clustering based classification for out-of-domain documents. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 210–219, 2007.

[42] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proc. of the International Conference on Machine Learning (ICML'07)*, pages 193–200, 2007.

[43] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

[44] T. Dietterich. Ensemble methods in machine learning. In *Proc. of the International Workshop on Multiple Classifier Systems (MCS'00)*, pages 1–15, 2000.

[45] A. Doan and A. Y. Halevy. Semantic-integration research in the database community. *AI Magazine*, 26:83–94, 2005.

[46] G. Dong and J. Li. Efficient mining of emerging patterns: discovering trends and differences. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 43–52, 1999.

[47] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *Proc. of the International Conference on Machine Learning (ICML'09)*, pages 289–296, 2009.

[48] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proc. of the International Conference on Machine Learning (ICML'00)*, pages 255–262, 2000.

[49] W. Fan. Systematic data selection to mine concept-drifting data streams. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 128–137, 2004.

[50] W. Fan and I. Davidson. On sample selection bias and its efficient correction via model averaging and unlabeled examples. In *Proc. of the SIAM International Conference on Data Mining (SDM'07)*, pages 320–331, 2007.

[51] W. Fan, E. Greengrass, J. McCloskey, P. S. Yu, and K. Drummey. Effective estimation of posterior probabilities: Explaining the accuracy of randomized decision tree approaches. In *Proc. of the IEEE International Conference on Data Mining (ICDM'05)*, pages 154–161, 2005.

[52] W. Fan, P. S. Yu, and H. Wang. Mining extremely skewed trading anomalies. In *Proc. of the International Conference on Extending Database Technology (EDBT'04)*, pages 801–810, 2004.

[53] J. Farquhar, D. Hardoon, H. Meng, J. Shawe-taylor, and S. Szedmak. Two view learning: Svm-2k, theory and practice. In *Advances in Neural Information Processing Systems (NIPS'05)*, pages 355–362, 2005.

[54] X. Z. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proc. of the International Conference on Machine Learning (ICML'04)*, pages 281–288, 2004.

[55] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[56] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 3(2):916–954, 2008.

[57] K. Ganchev, J. Graca, J. Blitzer, and B. Taskar. Multi-view learning over structured and non-identical outputs. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI'08)*, pages 204–211, 2008.

[58] J. Gao, H. Cheng, and P.-N. Tan. A novel framework for incorporating labeled examples into anomaly detection. In *Proc. of the SIAM International Conference on Data Mining (SDM'06)*, pages 594–598, 2006.

[59] J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu. Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing, Special Issue on Data Stream Management*, 12(6):37–49, 2008.

[60] J. Gao, W. Fan, and J. Han. On appropriate assumptions to mine data streams: Analysis and practice. In *Proc. of the IEEE International Conference on Data Mining (ICDM'07)*, pages 143–152, 2007.

[61] J. Gao, W. Fan, J. Han, and P. S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proc. of the SIAM International Conference on Data Mining (SDM'07)*, pages 3–14, 2007.

[62] J. Gao, W. Fan, J. Jiang, and J. Han. Knowledge transfer via multiple model local structure mapping. In *Proc. of the the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 283–291, 2008.

[63] J. Gao, W. Fan, Y. Sun, and J. Han. Heterogeneous source consensus learning via decision propagation and negotiation. In *Proc. of the the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 339–347, 2009.

[64] J. Gao, W. Fan, D. Turaga, S. Parthasarathy, and J. Han. A spectral framework for detecting inconsistency across multi-source object relationships. In *Proc. of the IEEE International Conference on Data Mining (ICDM'11)*, to appear, 2011.

[65] J. Gao, W. Fan, D. Turaga, O. Verscheure, X. Meng, L. Su, and J. Han. Consensus extraction from heterogeneous detectors to improve performance over network traffic anomaly detection. In *Proc. of the IEEE International Conference on Computer Communications Mini-Conference*, pages 181–185, 2011.

[66] J. Gao, G. Jiang, H. Chen, and J. Han. Modeling probabilistic measurement correlations for problem determination in large-scale distributed systems. In *Proc. of the International Conference on Distributed Computing Systems (ICDCS'09)*, pages 623–630, 2009.

[67] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. Graph-based consensus maximization among multiple supervised and unsupervised models. In *Advances in Neural Information Processing Systems (NIPS'09)*, pages 585–593, 2009.

[68] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. A graph-based consensus maximization approach for combining multiple supervised and unsupervised models. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, preprint, 2011.

[69] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On community outliers and their efficient detection in information networks. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 813–822, 2010.

[70] R. Ge, M. Ester, B. J. Gao, Z. Hu, B. Bhattacharya, and B. Ben-Moshe. Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. *ACM Transactions on Knowledge Discovery from Data*, 2(2):1–35, 2008.

[71] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis* (2nd ed.). Chapman and Hall, 2004.

[72] A. Genkin, D. D. Lewis, and D. Madigan. Bbr: Bayesian logistic regression software. http://stat.rutgers.edu/~madigan/BBR/.

[73] L. Getoor and B. Taskar. *Introduction to statistical relational learning.* MIT Press, 2007.

[74] J. Ghosh and A. Acharya. Cluster ensembles. *WIREs Data Mining and Knowledge Discovery*, 1:305–315, 2011.

[75] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):4:1–4:30, 2007.

[76] M. Girvan and M. Newman. Community structure in social and biological networks. 99(12):7821–7826, 2002.

[77] S. Goil, H. Nagesh, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. In *Technical Report*, Department of Electrical and Computer Engineering, Northwestern University, 1999.

[78] A. Goldberg and X. Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proc. of HLT-NAACL 2006 Workshop on Textgraphs*, 2006.

[79] Z. Guo, G. Jiang, H. Chen, and K. Yoshihira. Tracking probabilistic correlation of monitoring data for fault detection in complex systems. In *Proc. of the International Conference on Dependable Systems and Networks (DSN'06)*, pages 259–268, 2006.

[80] A. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proc. of the the International Conference on Very Large Data Bases*, pages 9–16, 2006.

[81] D. Hall and S. McMullen. *Mathematical Techniques in Multisensor Data Fusion.* Artech House, 2 edition, 2004.

[82] J. Han and M. Kamber. *Data Mining: Concepts and Techniques* (2nd ed.). Morgan Kaufmann, 2006.

[83] D. R. Hardoon, S. R. Szedmak, and J. R. Shawe-taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16:2639–2664, 2004.

[84] T. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(4):784–796, 2003.

[85] S. Hirose, K. Yamanishi, T. Nakata, and R. Fujimaki. Network anomaly detection based on eigen equation compression. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 1185–1194, 2009.

[86] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky. Bayesian model averaging: a tutorial. *Statistical Science*, 14(4):382–417, 1999.

[87] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems (NIPS'06)*, pages 601–608. 2006.

[88] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 97–106, 2001.

[89] T. Idé and H. Kashima. Eigenspace-based anomaly detection in computer systems. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 440–449, 2004.

[90] T. Ide, A. Lozano, N. Abe, and Y. Liu. Proximity-based anomaly detection using sparse structure learning. In *Proc. of the SIAM International Conference on Data Mining (SDM'09)*, pages 97–108, 2009.

[91] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[92] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A survey. *ACM Computing Surveys*, 31(3):264–323, 1999.

[93] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 538–543, 2002.

[94] G. Jiang, H. Chen, and K. Yoshihira. Discovering likely invariants of distributed transaction systems for autonomic system management. *Cluster Computing*, 9(4):385–399, 2006.

[95] T. Joachims. Making large-scale svm learning practical. advances in kernel methods - support vector learning. *MIT-Press*, 1999.

[96] T. Joachims. Transductive learning via spectral graph partitioning. In *Proc. of the International Conference on Machine Learning (ICML'03)*, pages 290–297, 2003.

[97] U. Kang, B. Meeder, and C. Faloutsos. Spectral analysis for billion-scale graphs: Discoveries and implementation. In *Proc. of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'11)*, pages 13–25, 2011.

[98] G. Karypis. Cluto – family of data clustering software tools. http://glaros.dtc.umn.edu/gkhome/views/cluto.

[99] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[100] N. Khoa and S. Chawla. Robust outlier detection using commute time and eigenspace embedding. In *Proc. of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'10)*, pages 422–434, 2010.

[101] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proc. of the International Conference on Very Large Data Bases (VLDB'04)*, pages 180–191, 2004.

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

[103] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proc. of the International Conference on Machine Learning (ICML'00)*, pages 487–494, 2000.

[104] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, 2000.

[105] J. Kolter and M. Maloof. Using additive expert ensembles to cope with concept drift. In *Proc. of the International Conference on Machine Learning (ICML'05)*, pages 449–456, 2005.

[106] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proc. of the VLDB Endowment*, 3:484–493, 2010.

[107] F. Korn, S. Muthukrishnan, and Y. Wu. Modeling skew in data streams. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD'06)*, pages 181–192, 2006.

[108] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: Similarity measures and algorithms. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD'06)*, pages 802–803, 2006.

[109] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* John Wiley & Sons, 2004.

[110] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Computer Communication Review*, 35(4):217–228, 2005.

[111] R. Lehoucq, D. Sorensen, and C. Yang. Arpack users' guide: Solution of large-scale eigenvalue problems with implicitly restarted arnoldi methods. *SIAM Publications*, 1998.

[112] M. Lenzerini. Data integration: a theoretical perspective. In *Proc. of the the ACM Symposium on Principles of Database Systems*, pages 233–246, 2002.

[113] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate?: cross-domain collaborative filtering for sparsity reduction. In *Proc. of the International Jont Conference on Artifical Intelligence (IJCAI'09)*, pages 2052–2057, 2009.

[114] H. Li, Z. Nie, W.-C. Lee, L. Giles, and J.-R. Wen. Scalable community discovery on textual data with relations. In *Proc. of the ACM Conference on Information and Knowledge Management (CIKM'08)*, pages 1203–1212, 2008.

[115] S. Z. Li. *Markov random field modeling in computer vision.* Springer-Verlag, 1995.

[116] T. Li, C. Ding, and M. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *Proc. of the IEEE International Conference on Data Mining (ICDM'07)*, pages 577–582, 2007.

[117] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Proc. of the ACM Internet Measurement Conference (IMC'06)*, pages 147–152, 2006.

[118] X. Li and J. Bilmes. A Bayesian divergence prior for classifier adaptation. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS'07)*, pages 275–282, 2007.

[119] Z. Li, J. Liu, and X. Tang. Constrained clustering via spectral regularization. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 421–428, 2009.

[120] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proc. of the IEEE International Conference on Data Mining (ICDM'08)*, pages 413–422, 2008.

[121] B. Long, P. Yu, and Z. Zhang. A general model for multiple view unsupervised learning. In *Proc. of the SIAM International Conference on Data Mining (SDM'08)*, pages 822–833, 2008.

[122] B. Long, Z. Zhang, and P. S. Yu. Combining multiple clusterings by soft correspondence. In *Proc. of the IEEE International Conference on Data Mining (ICDM'05)*, 2005.

[123] B. Long, Z. M. Zhang, and P. S. Yu. A probabilistic framework for relational clustering. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 470–479, 2007.

[124] P. Luo, F. Zhuang, H. Xiong, Y. Xiong, and Q. He. Transfer learning from multiple source domains via consensus regularization. In *Proc. of the ACM Conference on Information and Knowledge Management (CIKM'08)*, pages 103–112, 2008.

[125] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[126] M. Markou and S. Singh. Novelty detection: a review-part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497, 2003.

[127] M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 23(6):859–874, 2011.

[128] M. M. Masud, T. Al-Khateeb, K. W. Hamlen, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham. Cloud-based malware detection for evolving data streams. *ACM Transactions on Management Information Systems*, 2(3):16:1–16:27, 2011.

[129] M. M. Masud, C. Woolam, J. Gao, L. Khan, J. Han, K. W. Hamlen, and N. C. Oza. Facing the reality of data stream classification: Coping with scarcity of labeled data. *Knowledge and Information Systems (KAIS)*, preprint, 2011.

[130] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.

[131] M. A. Munawar, M. Jiang, and P. A. S. Ward. Monitoring multi-tier clustered systems with invariant metric relationships. In *Proc. of the International Workshop on Software Engineering for Adaptive and Self-managing Systems*, pages 73–80, 2008.

[132] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proc. of the ACM Conference on Information and Knowledge Management (CIKM'00)*, pages 86–93, 2000.

[133] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 631–636, 2003.

[134] O. Okun and G. Valentini. *Supervised and Unsupervised Ensemble Methods and their Applications*. Springer, 2008.

[135] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[136] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2006.

[137] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[138] W. Punch, A. Topchy, and A. K. Jain. Clustering ensembles: Models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1866–1881, 2005.

[139] N. Quadrianto and C. H. Lampert. Learning multi-view neighborhood preserving projections. In *Proc. of the the International Conference on Machine Learning (ICML)*, pages 425–432, 2011.

[140] C. Ramachandran, R. Malik, X. Jin, J. Gao, K. Nahrstedt, and J. Han. Videomule: A consensus learning approach to multi-label classification from noisy user-generated videos. In *Proc. of the ACM International Conference on Multimedia (ACM MM'09)*, pages 721–724, 2009.

[141] A. Ross and A. Jain. Information fusion in biometrics. *Pattern Recognition Letters*, 24:2115–2125, 2003.

[142] D. M. Roy and L. P. Kaelbling. Efficient bayesian task-level transfer learning. In *Proc. of the International Jont Conference on Artifical Intelligence (IJCAI'07)*, pages 2599–2604, 2007.

[143] F. Salfner and M. Malek. Using hidden semi-markov models for effective online failure prediction. In *Proc. of IEEE Symposium on Reliable Distributed Systems (SRDS'07)*, pages 161–174, 2007.

[144] S. Satpal and S. Sarawagi. Domain adaptation of conditional probability models via feature subsetting. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'07)*, pages 224–235, 2007.

[145] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[146] M. Scholz and R. Klinkenberg. An ensemble classifier for drifting concepts. In *Proc. of the ECML/PKDD05 Workshop on Knowledge Discovery in Data Streams*, pages 53–64, 2005.

[147] G. Seni and J. Elder. *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*. Morgan & Claypool, 2010.

[148] S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 371–376, 2001.

[149] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[150] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[151] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proc. of the ICML'05 workshop on Learning with Multiple Views*, 2005.

[152] V. Singh, L. Mukherjee, J. Peng, and J. Xu. Ensemble clustering using semidefinite programming. In *Advances in Neural Information Processing Systems (NIPS'07)*, pages 1353–1360, 2007.

[153] X. Song, M. Wu, C. Jermaine, and S. Ranka. Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):631–645, 2007.

[154] A. Storkey and M. Sugiyama. Mixture regression for covariate shift. In *Advances in Neural Information Processing Systems (NIPS'06)*, pages 1337–1344. 2006.

[155] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.

[156] L. Su, Y. Yang, B. Ding, J. Gao, T. F. Abdelzaher, and J. Han. Hierarchical aggregate classification with limited supervision for data reduction in wireless sensor networks. In *Proc. of the ACM International Conference on Embedded Networked Sensor Systems (Sensys'11)*, pages 40–53, 2011.

[157] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Proc. of the IEEE International Conference on Data Mining (ICDM'05)*, pages 418–425, 2005.

[158] P. Sun and S. Chawla. On local spatial outliers. In *Proc. of the IEEE International Conference on Data Mining (ICDM'04)*, pages 209–216, 2004.

[159] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[160] A. Vinueza and G. Grudic. Unsupervised outlier detection and semi-supervised learning. Technical Report CU-CS-976-04, University of Colorado at Boulder, 2004.

[161] B. Wang, J. Tang, W. Fan, S. Chen, Z. Yang, and Y. Liu. Heterogeneous cross domain ranking in latent space. In *Proc. of the ACM conference on Information and Knowledge Management (CIKM'09)*, pages 987–996, 2009.

[162] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 226–235, 2003.

[163] H. Wang, H. Shan, and A. Banerjee. Bayesian cluster ensembles. In *Proc. of the SIAM International Conference on Data Mining (SDM'09)*, pages 211–222, 2009.

[164] P. Wang, C. Domeniconi, and K. B. Laskey. Nonparametric bayesian clustering ensembles. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'10)*, pages 435–450, 2010.

[165] X. Wang and I. Davidson. Discovering contexts and contextual outliers using random walks in graphs. In *Proc. of the IEEE International Conference on Data Mining (ICDM'09)*, pages 1034–1039, 2009.

[166] X. Wang and I. Davidson. Flexible constrained spectral clustering. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*, pages 563–572, 2010.

[167] X. Wang, N. Mohanty, and A. McCallum. Group and topic discovery from relations and their attributes. In *Advances in Neural Information Processing Systems (NIPS'06)*, pages 1449–1456, 2006.

[168] G. I. Webb, S. Butler, and D. Newlands. On detecting differences between groups. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 256–265, 2003.

[169] G. Weiss and F. Provost. The effect of class distribution on classifier learning. In *Technical Report ML-TR-43*, Rutgers University, 2001.

[170] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[171] W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.

[172] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[173] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[174] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 824–833, 2007.

[175] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 927–936, 2009.

[176] Y. Yang, X. Wu, and X. Zhu. Combining proactive and reactive predictions for data streams. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05)*, pages 710–715, 2005.

[177] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796 – 808, 2008.

[178] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proc. of the International Conference on Machine Learning (ICML'04)*, pages 114–121, 2004.

[179] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted hmms for unusual event detection. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 611–618, 2005.

[180] K. Zhang, W. Fan, X. Yuan, I. Davidson, and X. Li. Forecasting skewed biased stochastic ozone days: Analyses and solutions. In *Proc. of the IEEE International Conference on Data Mining (ICDM'06)*, pages 753–764, 2006.

[181] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*, 20(1):45–57, 2001.

[182] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS'04)*, pages 321–328, 2004.

[183] D. Zhou and C. Burges. Spectral clustering and transductive learning with multiple views. In *Proc. of the International Conference on Machine Learning (ICML'07)*, pages 1159–1166, 2007.

[184] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *Advances in Neural Information Processing Systems (NIPS'03)*, pages 169–176, 2003.

[185] Z. Zhou, D. Zhan, and Q. Yang. Semi-supervised learning with very few labeled training examples. In *Proc. of the National Conference on Artificial Intelligence (AAAI'07)*, pages 675–680, 2007.

[186] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of the International Conference on Machine Learning (ICML'03)*, pages 912–919, 2003.

[187] X. Zhu and A. B. Goldberg, editors. *Introduction to Semi-Supervised Learning (Synthesis Lectures on Artificial Intelligence and Machine Learning)*. Morgan & Claypool, 2009.

[188] X. Zhu, R. Jin, Y. Breitbart, and G. Agrawal. Mmis07, 08: mining multiple information sources workshop report. *ACM SIGKDD Explorations Newsletter*, 10(2):61–65, 2008.

[189] A. Zien, U. Brefeld, and T. Scheffer. Transductive support vector machines for structured variables. In *Proc. of the International Conference on Machine Learning (ICML'07)*, pages 1183–1190, 2007.