Online learning for search and classification

Nguyen, Thanh Tam

2013

Nguyen, T. T. (2013). Online learning for search and classification. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/55287>

<https://doi.org/10.32657/10356/55287>

NANYANG
TECHNOLOGICAL
UNIVERSITY

**ONLINE LEARNING FOR SEARCH AND
CLASSIFICATION**

NGUYEN THANH TAM

SCHOOL OF COMPUTER ENGINEERING

2013

ONLINE LEARNING FOR SEARCH AND CLASSIFICATION

NGUYEN THANH TAM

School of Computer Engineering

A thesis submitted to the Nanyang Technological University

in fulfilment of the requirement for the degree of

Doctor of Philosophy

2013

Acknowledgment

I would like to express my sincere appreciation to the following organizations and people for their support and contribution:

Nanyang Technological University, Singapore, for granting me the opportunity and scholarship to undertake the Ph.D. programme and for providing me the most conducive research environment in the Centre for Advanced Information Systems for my research activities.

My two supervisors, Assistant Professor Kuiyu Chang and Associate Professor Siu Cheung Hui, for their patient guidance in my research and in the preparation for this thesis. They have given valuable advice and inspiration on my research, which makes research a truly joyful experience.

I would like to thank my friends, Mr. Guangxia Li, Mr. Zhen Hai, and Mr. Peilin Zhao for their insightful discussions.

Last but not least, I thank my wife Na Uy, who has supported me whole-heartedly throughout the duration of my studies.

Thank you all!

Abstract

Online learning is a common and useful tool for machine learning and data mining. In contrast to batch learning, online learning receives a sequence of training instances and uses some of them at a time. By the nature of online learning, the training instances may be processed only once. Therefore online learning algorithms can work on big data beyond the memory or disk capacity as well as streaming data. Moreover in document classification, online linear learning has been shown to be much more efficient than non-linear learning in terms of training and testing time. Therefore, online linear learning has recently become an active research topic.

This thesis proposes a research framework that attempts to solve the search and classification problems based on the online linear learning approaches. Specifically, we have proposed online learning classification algorithms that are able to work on multiple view datasets and an online learning-to-rank algorithm that improves the accuracy of a search engine. The main research contributions are listed as follows:

- Feature selection. We have investigated a number of newly supervised term weighting methods to improve the performance of text classification. These methods are evaluated on a number of text datasets and compared with other well-known unsupervised and supervised term weighting methods including $tf \times idf$ and $tf \times rf$.
- Online classification. We have proposed several online learning algorithms that can be used for topic classification. The proposed online algorithms were shown to outperform existing online learning algorithms on benchmarked datasets such as letter recognition and sentiment analysis.
- Two-view online learning. We have proposed a two-view online learning algorithm, which can work on two-view datasets. The algorithm was evaluated on multiple view datasets such as Web page classification and math topic classification. The experimental results shown that it worked better than other competitors did.

- Online learning-to-rank. For search engine, we have proposed an online learning-to-rank algorithm, which was to learn a scoring function to re-rank the search result. The proposed algorithm improved the accuracy of a search system on a math document dataset.

In summary, our proposed approaches have been benchmarked against competing algorithms, outperforming them on numerous real-life datasets. However, these are only preliminary successes. For future work, we will continue with our investigation on feature selection, online classification, multiple view online learning, and online learning-to-rank.

Contents

Abstract

Acknowledgment ii

1 Introduction 1

1.1	Background and Motivation	1
1.1.1	Term Weighting	3
1.1.2	Online Linear Learning	3
1.2	Project Objectives	5
1.3	Project Scope and Limitations	6
1.4	Approach and Methodology	6
1.4.1	Feature Selection	7
1.4.2	Online Linear Classification	8
1.4.3	Online Learning to Rank	8
1.5	Contributions	9
1.6	Organization of the Thesis	10

2 Related Work 11

2.1	Term Weighting Method	11
2.1.1	Unsupervised Term Weighting	12
2.1.2	Supervised Term Weighting	12
2.1.3	Linearity Analysis	14
2.2	Text Categorization	15
2.2.1	Machine Learning Classifiers	16
2.2.2	None Vector Space Model Approach	17
2.2.3	Centroid-based Classifiers	17
2.2.4	Discussion	17

2.3	Perceptron-based Online Learning	18
2.3.1	Online Learning Problem Setting	18
2.3.2	Online Passive Aggressive Learning	18
2.3.3	Distribution-aware Online Learning	19
2.3.4	Two-view Online Learning	20
2.3.5	Online Learning-to-Rank	21
2.3.6	Discussion	21
2.4	Summary	22
3	Kullback-Leibler Term Weighting	23
3.1	Term Weighting Formulation	24
3.1.1	Impurity Measures	25
3.1.2	Kullback-Leibler Divergence Measure	26
3.1.3	Comparing $tf \times rf$ and $tf \times KL$	27
3.1.4	Generalized $tf \times KL$	29
3.2	Performance Evaluation	30
3.2.1	Datasets	30
3.2.2	Performance Comparison	31
3.2.3	When to Use $tf \times KL$?	32
3.2.4	Performance on Imbalanced Datasets	33
3.3	Summary	36
4	Centroid-based Classifiers	38
4.1	Centroid-based Classification	39
4.1.1	Word Cloud Model	41
4.1.2	Centroid-based Classifier with Kullback-Leibler Divergence	43
4.1.3	Growth of Class Document Frequency	45
4.1.4	Centroid-based Classifier with Jensen-Shannon Divergence	46
4.2	Performance Evaluation	49
4.2.1	Datasets	49
4.2.2	Term Weighting Visualization	49
4.2.3	Binary Classification	52
4.2.4	One-off Classification of Multi-class Imbalanced Data	56
4.2.5	$tf \times JS$ versus $tf \times KL$ on Imbalanced Binary Datasets	58

4.3	Summary	61
5	Online Linear Learning	62
5.1	Passive Aggressive Mahalanobis	63
5.1.1	Hard Margin PAM	63
5.1.2	Soft Margin PAM	63
5.1.3	Covariance Matrix Estimation	65
5.1.4	PAM Error Analysis	66
5.2	Performance Evaluation	67
5.2.1	Datasets	67
5.2.2	Cumulative Error Rate	68
5.2.3	Classification Accuracy	70
5.2.4	Online Microblog Data	70
5.3	Summary	72
6	Two-View Online Linear Learning	73
6.1	Math Feature Extraction	74
6.2	Two-view Passive Aggressive Algorithm	75
6.2.1	Relationship between Views	75
6.2.2	Getting Rid of Parameter η	77
6.3	Performance Evaluation	78
6.3.1	Datasets	78
6.3.2	Two-view Learning Evaluation	78
6.3.3	Math Topic Classification	79
6.3.4	View Weight Parameter Learning	82
6.4	Summary	82
7	Online Learning to Rank	84
7.1	Math Document Preprocessing	85
7.1.1	Math Document Representation	86
7.1.2	Math Document Feature Extraction	86
7.2	Data Model	87
7.2.1	Matching Model	87
7.2.2	Ranking Model	88
7.3	Performance Evaluation	92

7.3.1	Dataset	92
7.3.2	Experimental Setup	93
7.3.3	Math Feature Extraction Evaluation	93
7.3.4	Ranking Algorithm Evaluation	96
7.3.5	Retrieval Performance Evaluation	96
7.4	Summary	98
8	Conclusion and Future Work	99
8.1	Summary of the Thesis	99
8.2	Future Work	101
8.2.1	Supervised Term Weighting	101
8.2.2	Online Multi-class Learning	101
8.2.3	Math Topic Classification	102
A	List of Publications	103
A.1	Journal Papers	103
A.2	Conference and Workshop Papers	103
	Bibliography	104

List of Tables

2.1	Notations	11
2.2	Supervised Term Weighting Methods	12
2.3	Linearity of Supervised Term Weighting Methods.	15
3.1	Supervised Term Weighting Schemes.	28
3.2	$tf \times rf$ vs. $tf \times KL$ (* denotes non-zero value).	29
3.3	Datasets (* denotes multi-class).	30
3.4	5-fold Cross Validation Accuracy.	31
4.1	Term Counts (C_1, C_2) and Centroids ($\mathbf{c}_1, \mathbf{c}_2$) for an Extreme Example	40
4.2	Centroid Growth versus Class Document Frequency	46
4.3	Datasets (* denotes multi-class)	49
4.4	Imbalanced 20-Newsgroup and Reuters-21578 Datasets	57
5.1	Datasets.	68
5.2	F1 (%) for Positive (+) and Negative (-) Classes.	70
6.1	Summary of Datasets in Our Experiments	78
6.2	F-Measure on 3 Benchmark Datasets	78
6.3	Math Overflow Sample Documents	80
6.4	F-Measure Comparison on Math Overflow Datasets (* trained on the Math & Text Dataset, with results for individual views shown)	82
6.5	The Adaptive Two-view PA Results for All Datasets	82
7.1	Textual Math and Math Feature Properties	95
7.2	Metadata Properties (* using Equation (7.5)).	97

List of Figures

1.1	Proposed Research Framework	7
3.1	The Effect of Labeled Data based across Domains.	33
3.2	F1, Precision, Recall for Positive (+) and Negative (-) Classes on Imbalanced Dataset.	34
3.3	Improvement of $tf \times KL$ over Runner-up method for Positive Class.	35
3.4	Improvement of $tf \times KL$ over Runner-up method for Negative Class.	36
4.1	Word Clouds	41
4.2	IDF Term Weight Visualization of the Movie Review Dataset	50
4.3	RF Term Weight Visualization of the Movie Review Dataset	50
4.4	KL Term Weight Visualization of the Movie Review Dataset	51
4.5	JS Term Weight Visualization of the Movie Review Dataset	51
4.6	Movie Review Dataset Results	52
4.7	Sentiment Polarity Dataset Results	53
4.8	F-Measure for Multi-domain Sentiment Dataset	54
4.9	Zero Weighted Terms (%) in each Centroid for the Multi-domain Sentiment Dataset	55
4.10	Evolution of Centroid Term “like”	56
4.11	Evolution of Centroid Term “movie”	56
4.12	F-Measure for Imbalanced 20-Newsgroup Dataset	57
4.13	F-Measure for Imbalanced Reuters-21578 Dataset	58
4.14	F1, Precision, Recall for Positive Class on the Book Domain	58
4.15	F1, Precision, Recall for Positive Class on the DVD Domain	58
4.16	F1, Precision, Recall for Positive Class on the Electronics Domain	59
4.17	F1, Precision, Recall for Positive Class on the Kitchen Domain	59
4.18	F1, Precision, Recall for Negative Class on the Book Domain	60

4.19	F1, Precision, Recall for Negative Class on the DVD Domain	60
4.20	F1, Precision, Recall for Negative Class on the Electronics Domain	60
4.21	F1, Precision, Recall for Negative Class on the Kitchen Domain	60
5.1	Cumulative Error Rate for BUPA and CRX.	68
5.2	Cumulative Error Rate for USPS and MNIST.	69
5.3	Cumulative Error Rate for WebKB.	69
5.4	Cumulative Error Rate for the Twitter Sentiment Dataset.	71
7.1	Math-Aware Search Engine Framework	85
7.2	Math-Aware Search Engine	86
7.3	FSM for Extracting Textual Math Features	94
7.4	Performance Comparison on the Math Expression Dataset	95
7.5	Cumulative Error Rate on the Math Overflow Dataset	96
7.6	Performance Comparison on the Math Overflow Dataset	97
8.1	Thesis Summary	99

Chapter 1

Introduction

1.1 Background and Motivation

Since its introduction back in the nineties, the World Wide Web has become the de facto data repository of modern human knowledge. The Web today contains different types of data and information including texts, images, videos, hyper-links, sentiments, mathematical expressions, etc. These data can be analyzed to uncover nuggets of knowledge ranging from competitive intelligence for corporations, opinions about public services for governments, product sentiments for manufacturers, to mathematical documents for students and scientists. Especially, user-generated data comes from social networks, that store many kinds of rich data such as user profiles, photos, videos, comments, etc. Moreover, with the phenomenal growth of general purpose search engines like Google ¹, Bing ², Yahoo ³, etc., Internet users are now spoiled for choices insofar as searching for simple information. Existing search engines were largely developed for the early years of the Internet, and are gradually showing signs of age, due to the increasingly demanding information needs of today's users. In fact, current search engines do not really understand a user's true information need. For instance, if we enter the query "google", Google simply returns all Web pages containing the term "google". It does not understand that we might be searching for the "Google search engine" unless it is explicitly keyed in.

To overcome this shortcoming, semantic search engines have been proposed with the goal of improving search accuracy by understanding the user's need [49]. Caching techniques have also been used to increase the precision of semantic search results [75]. In addition, semantic

¹<http://www.google.com>

²<http://www.bing.com>

³<http://search.yahoo.com>

search engines offer extra relevant information such as related search, reference results, etc. For example, if we enter the query “google”, a semantic search engine will not only return all Web pages related to Google, but also offer us options like “You may also wish to search for: Hakia, Yahoo, AOL Search, MSN Search, Ask.com”, which are names of other popular search engines. In this case, the semantic search engine may correctly guess that we are searching for a search engine and thus suggests other search engines. Apart from related search and reference results, semantic search engines can also “boost accuracy by taming ambiguity via an understanding of context”⁴.

There are many specialized domains for which a semantic search engine makes more intuitive sense than a general purpose search engine like Google. The semantic meaning in each domain is largely different from one another. Thus, a general semantic search engine will not be suitable for all domains. Taking sentiment and social media analysis as an example, the figures and facts about the recent growth of social network come to the conclusion that social network users becomes denser over the years. Currently, it reaches nearly one fourth of population around the world. Compared to 2012, the social network audience grows 18% in 2013⁵. Subsequently, the Asia-Pacific becomes the area with largest population. Taking Singapore as an example, there are 2.9 million people using Internet and 94.3% of them spend 4.4 hours per day on social networks⁶. It means that if we meet two people in Singapore, one of them is visiting social networks. Hence, Singapore is one of the most crowded social network users. But to understand the emotion of Singaporeans or what they are talking about are still an open problem.

Large number of users have been generating a vast amount of data over the time. Let's take the top social networks such as Facebook⁷, Twitter⁸, and Youtube⁹ as an example, users from these system have been generate a vast of data everyday. There are 400 million tweets sent daily in Twitter¹⁰. Youtube users have been uploading 144,000 hours of video daily¹¹. And 4.75 billion pieces of content are shared on Facebook every day¹². Together with

⁴<http://www.informationweek.com/news/showArticle.jhtml?articleID=222400100>

⁵<http://www.emarketer.com/Article/Social-Networking-Reaches-Nearly-One-Four-Around-World/1009976>

⁶<http://www.ida.gov.sg/~media/Files/InfocommLandscape/Technology/TechnologyRoadmap/SocialMedia.pdf>

⁷<http://www.facebook.com>

⁸<http://www.twitter.com>

⁹<http://www.youtube.com>

¹⁰http://articles.washingtonpost.com/2013-03-21/business/37889387_1_tweets-jack-dorsey-twitter

¹¹<http://www.youtube.com/yt/press/statistics.html>

¹²<https://www.facebook.com/photo.php?fbid=10151908376716729&set=a.10151908376636729.1073741825.20531316728&type=3&theater>

the development of mobile devices, it has never been easier to share and access information anywhere and anytime. Therefore, the user-generated data is not only in high dimension but also big volume, which is the challenge for data scientists. Normally, to apply classification techniques into these kinds of data, one commonly has to use the following methods:

- *Term weighting.* In this methods, text data in bag-of-word model will be converted into numeric vectors, where each document will be represented by a vector in the vector space model.
- *Online linear learning.* Generally, the objective of this method is to learn a linear model based on the training data in the bag-of-word model.

1.1.1 Term Weighting

While working with text data, especially in document classification and natural language processing (NLP), one has to convert the textual data into numerical vectors in high dimensional spaces. Normally, the bag-of-word model is often used, where term weighting techniques are applied to generate feature vectors. In this model, documents are represented by multi-sets of terms. And they are then converted into vectors based on the appearance of terms. Some term weighting methods commonly used can be listed as term occurrence (binary), term frequency (tf), document frequency (df), or their scaled versions such as $\log(tf)$. While term occurrence and frequency represent whether a term appear in a documents, they fail to express the importance of the term in the whole collection. On the other hand, df represents how many documents contain a term. This quantitative variable is very useful to determine the role of term in the collection. Furthermore, in bag-of-word model, each document notably tends to contain several stop words such as “the”, “to”, etc. Hence, their less discriminative value, stop words should have smaller weight. The combination of tf and inverse df is commonly used in practice. Since these methods ignore the term distribution, they are classified as unsupervised term weighting methods. Recently, there are a few supervised term weighting methods [45, 51, 58, 63], which take into consideration the distribution of term across classes to determine the weights. The experimental results show that supervised term weighting methods outperform their unsupervised competitors.

1.1.2 Online Linear Learning

Due to the nature of term weighting techniques, where the number of words in practice is very large, the presenting vectors are commonly in extremely high dimension. Linear classification

is such a popular and useful tool document classification. In this application, one can use linear classification techniques such as support vector machines (SVM) [18, 12] and logistic regression (LR) [20] to process the data. In contrast to non-linear classifiers, which map data from low to high dimension such as kernel methods, linear counterparts work on the original input data. Since non-linear classifiers work on higher dimension space, they have high accuracy than that of the linear approaches, especially in inseparable data. Although linear classifiers fail to work on this kind of data, they have been shown to have competitive performances in high dimension data such as document data with the non-linear ones. In addition, the training and testing procedures in linear classification are much more efficient. Therefore, the research on linear classification together with social media analysis is currently a very active topic.

However, the above algorithms work under the assumption that the whole training data can be loaded into the computer memory so there should be another class of algorithms which can work on the large-scale setting. For instance, in practice, the size of training data may be larger than the memory. Even if the training data can fit the memory capacity, these algorithms may become very slow due to the frequent disk access [89]. There are different ways to solve this problem. For example, one can use distributed computing frameworks such as Apache Hadoop¹³, Apache Spark¹⁴. The research in this area is still in its early infant stage. In this research, we mainly focus on online linear classification approaches, which work on rounds and receive training data in sequence and process some examples at each round. Since a part of training data is processed once, they can be trained on a training dataset which goes beyond the memory capacity.

The first kind of online linear learning algorithms is centroid-based classifiers, where the trained models are built based on the linear combination of training instances. In the simplest case, a centroid vector is the mean of all documents in a certain class. To improve the performance of the centroid-based classifier, Guan et al proposed class-feature-centroid classifier (CFC) [35]. This algorithm is based on the $tf \times idf$ term weighting technique to determine the prototype vectors representing for each class. The proposed algorithm has the training time linear to number of training instances and its performance is comparable to SVM's.

Since the CFC algorithms are based on term weighting techniques to determine their centroids, they are only work on the textual data. There is another class of linear classifiers

¹³<http://hadoop.apache.org/>

¹⁴<http://spark.incubator.apache.org/>

which are developed by solving convex optimization problem, which can work on other kinds of data such as letter recognition and image classification. The important objective is to develop fast optimization algorithms. One such popular online algorithm is the stochastic gradient descent method (SGD) [77]. While training the linear classifiers, we want to minimize the training loss but low training error may not imply that the testing accuracy will be high. In order to overcome this overfitting training problem, the regularization techniques have been proposed. The main idea of these methods is to push back the values of variables toward zero. L1 and L2 regularizations are two major techniques used in linear classification [68, 84], which introduces regularization terms in the objective function of the optimization problem. The passive aggressive algorithms proposed by Crammer et al [21] belong to L2 regularization and adapt large margin technique. They are shown to outperform other online learning competitors.

In practice, the data can be represented by different views so it will be useful to have classifiers which can work well on this kind of data. It is very easy for us to see this problem around such as Web page classification. A Web page contains not only normal text data but also other kinds of data such as links, images, etc. and all of them represent the Web page. A trivial approach is to combine them together to come up with one data instance only and then apply linear algorithms on the combined data. However, since we do not know which data views are better, concatenation several views may not get the optimal solution. Therefore, if we treat these views separately and take advantage of the quality of each view, we may improve the performance of a classifier.

In 1998, Blum and Mitchell proposed the co-training [8], which use the unlabelled sample to boost the performance of a learning algorithm. In their problem setting, each training instance consists of two distinct views that will be used to train two Bayes classifiers. The co-training classifier has been shown to outperform naive Bayes classifier on Web page classification about 6% in terms of error rate. Later, Farquhar et al. [31] proposed a large margin two-view SVM, which is an extension of SVM algorithm. These two algorithms have been shown to work well on Web page classification problem.

1.2 Project Objectives

The objective of this research is to investigate term weighting and online learning methods that can be applied to search and classification. Specifically, we will investigate the following issues.

- *Feature selection methods.* Historically, feature selection has played a very important role in the performance of classifiers and search systems. Several studies investigating feature selection methods have been carried out on text documents. Since in document classification, we have the class label of training documents. If we can use this information to calculate the term weights, we may improve the performance of the classifiers. The goal is to develop efficient and effective term weighting methods for document classification.
- *Online linear classification.* One of the most significant tasks in machine learning and data mining is to organize text documents. Currently, there are many machine learning methods developed for text categorization. However, they are not efficient for document classification because documents may include many views. And most of them are batch learning algorithms. Therefore, new online learning algorithms that work on multiple view data are needed.
- *Online learning-to-rank.* Since feature extraction in a few kinds of data is still in infancy stages, machine learning approach should be applied to this data. Take information retrieval as an example, that the feature set is not good enough can make ranking search results incorrect. The objective is to improve the performance of the search system by applying learning-to-rank algorithms.

1.3 Project Scope and Limitations

We will focus mainly on online learning approaches and their applications to search and classification, where document categorization datasets are selected to evaluate the learning algorithms. The primary objective is to investigate different fundamental information retrieval and machine learning methods needed to facilitate large scale and multiple view data. Text information retrieval and categorization have been investigated for a long time. Difficulties arise, however, when an attempt is made to implement classification and search systems. Fortunately, recent research shows that online learning works very efficient on large scale data. Therefore, this research will focus on online learning for search and classification.

1.4 Approach and Methodology

The major focus of this research is to develop online learning approaches for search and classification. The proposed algorithms should work on real-world documents containing

single or multiple views. Each view represents one kinds of data in the documents, where each document contains textual data. There should be a good term weighting method to convert this data into a feature vector. Therefore, we also investigated on supervised term weighting and math feature extraction methods. Finally, the proposed approaches were evaluated on document classification datasets such as sentiment analysis, Web page classification, and math topic classification.

Figure 1.1: Proposed Research Framework

Figure 1.1 shows our proposed research framework for search and classification. As shown in the figure, there are several challenges and research problems including feature selection, online classification, and information retrieval. We briefly describe each of the challenges in the following subsections.

1.4.1 Feature Selection

For text data, one can use the bag-of-words model and construct vectors corresponding to each document by using term weighting techniques. Many term weighting techniques such as tf and $tf \times idf$ [62] have been applied successfully in information retrieval as well as text categorization. However, these unsupervised techniques ignore the document class label, which can be very useful in improving the performance of search systems. By taking into account the class label, supervised term weighting techniques such as $tf \times rf$ [51] were shown to improve classification accuracy.

1.4.2 Online Linear Classification

Even though online learning algorithms often run faster than batch learning algorithms like SVM [18], their accuracy is typically lower than that of batch learning algorithms. Our goal is to develop a high-performance online learning algorithm for math topic classification and retrieval using two views. Since math documents contain both text and math, also known as two views, the new algorithms have to work on both views at the same time without one dominating over the other. Specifically, let $(\mathbf{x}_t^A, \mathbf{x}_t^B, y_t) \in \mathbb{R}^n \times \mathbb{R}^m \times \{-1, 1\}$ be the training data arriving in sequence, where $\mathbf{x}_t^A$ and $\mathbf{x}_t^B$ are text and math vectors representing the document t . The objective is to learn the weight vectors of the function $f(\mathbf{x}_t^A, \mathbf{x}_t^B) = \eta \mathbf{w}^A \cdot \mathbf{x}_t^A + (1 - \eta) \mathbf{w}^B \cdot \mathbf{x}_t^B$ so that f can predict the label y_t of $(\mathbf{x}_t^A, \mathbf{x}_t^B)$. In theory, multi-view learning algorithms can work on different views of data to achieve better performances than their single view counterparts.

To the best of our knowledge, there are no standard feature extraction methods for math documents. However, the performance of math search systems depends heavily on the feature extraction process. In order to improve the performance of math search systems, learning-to-rank algorithms can be applied to re-rank the search results. Let $score(\mathbf{d}_i, \mathbf{q}_j) = \mathbf{w} \cdot (\mathbf{d}_i \otimes \mathbf{q}_j)$ be the scoring function of document $\mathbf{d}_i$ and query $\mathbf{q}_j$. The problem is to learn a weight vector $\mathbf{w}$ such that if $\mathbf{q}_j$ is more relevant to $\mathbf{d}_i$ than $\mathbf{q}_k$, then $score(\mathbf{d}_i, \mathbf{q}_j) \geq score(\mathbf{d}_i, \mathbf{q}_k)$.

1.4.3 Online Learning to Rank

Math documents can be significantly more complex than text documents; it can involve mathematical expressions that are sequence sensitive in two dimensions, e.g., matrices, fractions, nested forms. Classical text retrieval methods, which typically throw away the linear sequence information in text, are thus very poorly suited to process math documents. Therefore, new retrieval methods are needed to handle mathematical documents. Furthermore, since math expressions are highly structured and complex, new ways are needed to meaningfully rank and display search results, with which users can easily navigate and interact. We now propose our math-aware information retrieval model as follows.

Let $\mathbb{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_n\}$ and $\mathbb{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_m\}$ be a collection of math documents and a set of user queries, respectively, where $\mathbf{d}_i$ and $\mathbf{q}_j$ contain both text and math data. The math information retrieval problem formulates a *ranking function* $f : \mathbb{D} \times \mathbb{Q} \mapsto \mathbb{R}$, which defines an order among documents based on the queries via a *framework* F that considers all math documents and queries. For example, in text information retrieval, the framework

F is typically the vector space of document and query points, with the ranking function determined by cosine similarity.

Generally, the cosine similarity should be replaced by the Mahalanobis distance. Since math features are not as well defined as text ones, the Mahalanobis matrix can be learnt from data by applying a learning-to-rank approach. Doing so, we can improve the retrieval precision of math search systems. Moreover, the role of text and math in math document is unlikely the same. Weighting each view optimally is another interesting research problem by its own right.

1.5 Contributions

Several novel methods have been proposed to achieve the objectives listed in Section 1.2. The contributions of this research can be summarized as follows:

- *Kullback-Leibler term weighting.* We proposed a term weighting scheme called $tf \times KL$ (term frequency Kullback-Leibler), which is proportional to the ratio of two class distributions, i.e., the Kullback-Leibler Divergence. In this research, we used it to improve the accuracy of classifiers in binary classification.
- *Centroid-based classifiers for text categorization.* We proposed new centroid-based classifiers in association with supervised term weighting methods for text categorization.
- *Mahalanobis passive-aggressive algorithms for online classification.* We proposed a family of Mahalanobis Passive-Aggressive algorithms, which incorporate the covariance of misclassified samples lying near the class boundary. The proposed algorithms can be used for online classification, especially text categorization.
- *Two-View online algorithm for online classification.* We proposed the online two-view learning algorithm, which works on multiple views of data at the same time. We then apply it for online math topic classification by combining the math and textual views of the data.
- *Online learning-to-rank.* To support searching both text and math, we proposed new matching and ranking models for math search. In the matching model, the search results are retrieved based on the intersection of text and math indices. Then, the learning-to-rank algorithm is applied to re-rank the search results.

1.6 Organization of the Thesis

The rest of the thesis is organized as follows:

- In Chapter 2, we conduct a literature review on the topics related to this research.
- In Chapter 3, we formally define the supervised term weighting problem and propose the KL term weighting approach.
- In Chapter 4, we introduce a novel centroid-based classification algorithm in association with supervised term weighting schemes for text categorization.
- In Chapter 5, we propose a family of distribution-aware online classification algorithms.
- In Chapter 6, we extend the online learning algorithm for the two-view learning problem and apply it to Web page and math topic classification.
- In Chapter 7, we adapt the online learning algorithm for the learning-to-rank problem and evaluate it on a math document dataset.
- In Chapter 8, we draw conclusions for this research as well as discuss some future directions.

Chapter 2

Related Work

2.1 Term Weighting Method

The Vector Space Model (VSM) was proposed for information retrieval systems by Salton [78], where each document is represented by a bag of words/terms vector, $\mathbf{w}_i = (w_{i1}, w_{i2}, \dots, w_{in})$. Generally, term weights are determined based on the local and global weights as follows:

$$w_{ij} = f(L_{ij}, G_j, N_i),$$

where L_{ij} is the local weight of term j in document i , G_j is the global weight of term j , N_i is a normalization factor, and f is an arbitrary function.

Depending on how the global weight is computed, term weighting can either be unsupervised or supervised. In unsupervised term weighting, the global weight is calculated independently of the class labels, while supervised term weighting computes the global weight from labeled data.

To review term weighting methods, we first define a few notations in Table 2.1, which will be used in subsequent term weighting equations.

Table 2.1: Notations

Notation	Explanation
tf	Term frequency
$n^+(x)$	Number of positive documents containing the term
$n^-(x)$	Number of negative documents containing the term
$n(x)$	Number of documents containing the term
N^+	Total number of positive documents
N^-	Total number of negative documents
N	Total number of documents in the dataset

2.1.1 Unsupervised Term Weighting

In unsupervised term weighting, the local weight L_{ij} is defined as the frequency of a term in a document (tf). In the binary case (also known as term presence), L_{ij} can be a binary value, i.e., $L_{ij} = 1$ if term j exists in the document i , otherwise $L_{ij} = 0$. In addition, there are also other term frequency models including $\log(tf) + 1$, $\log(tf + 1)$, and $tf / \max(tf)$.

For global weight, one of the most popular term weighting methods is the Inverse Document Frequency (idf) [62, 79], which measures the discriminatory power of a term, namely term specificity [44]. It is the ratio of the total number of documents in the collection to the number of documents containing the term. The idf has been applied successfully to many classical information retrieval tasks and categorization problems based on the ranking function BM25 or its variants like BM25t [34]. In practice, the product of tf and idf is used as follows:

$$tf \times idf(x) = tf \times \log \frac{N}{n(x)} = tf \times (\log N - \log n(x))$$

In this formula, tf and idf are the local and global weights of term x respectively. The local weight tf of a term says that it is more important if it occurs more frequently in a document. Intuitively, the global weight idf of a term is smallest if it occurs in every document (least discriminative) and largest if it appears in only one document (most discriminative).

2.1.2 Supervised Term Weighting

Apart from unsupervised term weighting, various supervised term weighting methods have been proposed with the goal of improving classification accuracy. These methods include $tf \times \chi^2$ (Chi squared), $DTWC - KL$ [45], $tf \times rf$ (Relevant Frequency) [51], WFO (Weighed Frequency and Odds) [58], and $tf \times \delta idf$ (delta idf) [63]. Table 2.2 lists some common supervised term weighting methods and their associated formulas.

Table 2.2: Supervised Term Weighting Methods

Method	Formula
$tf \times \chi^2$	$tf \times \frac{N[P(x,+)(1-P(x,-))-P(x,-)(1-P(x,+))]^2}{P(x)(1-P(x))P(+)P(-)}$
$tf \times rf$	$tf \times \log_2 \left(2 + \frac{n^+(x)}{\max(1, n^-(x))} \right)$
$tf \times \delta idf$	$tf \times \left(\log_2 \frac{N^-}{n^-(x)} - \log_2 \frac{N^+}{n^+(x)} \right)$
WFO	$\frac{n^+(x)}{N^+}^\lambda \left(\log \frac{n^+(x)}{n^-(x)} \frac{N^-}{N^+} \right)^{(1-\lambda)}$

In [2], Altincay and Erenel scrutinized six widely used term weighting methods expressed in terms of the positive and negative word probabilities. Based on their inter-relationships, these methods were then classified into two major groups, namely linear and nonlinear. For each of the linear and nonlinear category, the authors express the term weight as uniform functions of the positive and negative class probabilities.

Debole and Sebastiani [80, 24] proposed a new supervised term weighting scheme called $tf \times \chi^2$, which uses the Chi-square (χ^2) statistic to boost the weights of highly discriminative terms. Specifically, the idf weight is replaced by the χ^2 term statistic. The χ^2 value of a term is reduced if it is independent of the class and increased otherwise. The χ^2 value is used to quantify the discriminatory power of a term. It has been used in feature selection and classical statistical analysis. The $tf \times \chi^2$ method achieved 11% better performance than $tf \times idf$ on the Reuters-21578 dataset, which is a popular IR news articles dataset.

In [51], Lan et al. proposed another supervised term weighting method called $tf \times rf$, which is a two class extension to $tf \times idf$ whereby the idf component is replaced by rf (relevance frequency). Intuitively, $tf \times rf$ tries to give higher weightage to terms that appear more frequently in the positive class. However, there are some peculiarities with $tf \times rf$. First, $tf \times rf$ is asymmetric because it favors terms that appear more in the positive class only. This means that in practical two-class problems, the choice of which class to be “positive” will have an impact on term weighting and ultimately the classification performance. Second, $tf \times rf$ does not penalize terms that appear equally often in both classes. In both cases, terms are simply assigned their local tf weights. Similar to other supervised term weighting methods, $tf \times rf$ performed 3% better than $tf \times idf$ on the Ohsumed corpus, which is a medical abstract IR corpus.

In $tf \times \delta idf$ [63], Martineau and Finin replaced idf with the differential class inverse document frequency, which they named the delta inverse document frequency. The importance of a term is boosted if it is unevenly distributed between the positive and negative classes. The weight of a term equals to zero if it is evenly distributed among the two classes. The authors conducted experiments on a number of sentiment datasets and reported that $tf \times \delta idf$ performed 6% better than $tf \times idf$ on the publicly available movie review dataset.

The WFO [58] (Weighted Frequency and Odds) term weighting scheme is a radical approach that discards the local tf term completely. In addition, idf is replaced by a combination of the normalized document frequency and the odds ratio. An exponential parameter λ is used to weigh the document frequency and odds ratio; if $\lambda = 0$, WFO is comprised

entirely of the odds ratio, if $\lambda = 1$, WFO is simply the normalized document frequency. The WFO weight of a term approaches the maximum value if both its document frequency and class odds ratio are high. In general, WFO is a generalized version of $tf \times \delta idf$, where the parameter λ is used to favor either tf or δidf .

The above supervised term weighting techniques have been applied to various text categorization tasks including sentiment analysis [40, 13, 38], email spam filtering [60, 46], news classification, spyware detection [53], and job title classification [4].

2.1.3 Linearity Analysis

We analyze the linearity of three of the most popular term weighting schemes, $tf \times rf$, $tf \times \delta idf$, and WFO for binary classification problems. In all three term weighting methods, the consensus is that tf always plays an important role as a local weight. The other is the supervised term weighting factor which is the global weight. To compare these methods, we first define some terminologies as follows.

Let p^+ , p^- , and p be the probabilities of a term belonging to the positive class, negative class, and both classes respectively. We have $p^+ = \frac{n^+(x)}{N^+}$, $p^- = \frac{n^-(x)}{N^-}$, and $p = \frac{n}{N}$. In the following, the expressions of all term weights are derived based on p^+ and p^- .

In general, rf has the form

$$rf = \log_2 \left(2 + \frac{n^+(x)}{n^-(x)} \right)$$

Let $\alpha = 2^{rf}$ and replace $n^+(x)$ and $n^-(x)$ by N^+p^+ and N^-p^- in the above formula, we have

$$2 + \frac{N^+p^+}{N^-p^-} = \alpha \implies p^+ = (\alpha - 2) \frac{N^-}{N^+} p^- \quad (2.1)$$

From Equation (2.1), we can conclude that p^+ is linearly dependent on p^- or the term weight is a constant if the ratio p^+/p^- is unchanged.

For δidf , we have

$$\delta idf = \log_2 \frac{N^-}{n^-(x)} - \log_2 \frac{N^+}{n^+(x)}$$

Replacing $N^+/n^+(x)$ and $N^-/n^-(x)$ by $1/p^+$ and $1/p^-$, respectively, we have

$$\delta idf = \log_2 \frac{1}{p^-} - \log_2 \frac{1}{p^+} = \log_2 \frac{p^+}{p^-}$$

Let $\alpha = 2^{\delta idf}$, we have

$$\frac{p^+}{p^-} = \alpha \implies p^+ = \alpha p^- \quad (2.2)$$

Similar to rf , Equation (2.2) shows that δidf is a linear method as its weight depends on the ratio of p^+/p^- .

For WFO , we have

$$WFO = \frac{n^+(x)}{N^+}^\lambda \left(\log \frac{n^+(x)}{n^-(x)} \frac{N^-}{N^+} \right)^{(1-\lambda)}$$

Substituting p^+ and p^- into the above formula, we have

$$WFO = p^{+\lambda} \left(\log \frac{p^+}{p^-} \right)^{(1-\lambda)}$$

Let $\alpha = WFO^{\frac{1}{1-\lambda}}$, we have

$$\log \frac{p^+}{p^-} = \alpha p^{+\frac{\lambda}{\lambda-1}} \implies \frac{p^+}{p^-} = 2^{\alpha p^{+\frac{\lambda}{\lambda-1}}}$$

Let $\beta = 2^\alpha$, we have

$$\frac{p^+}{p^-} = \beta p^{+\frac{\lambda}{\lambda-1}} \implies p^- = \frac{p^+}{\beta p^{+\frac{\lambda}{\lambda-1}}} \quad (2.3)$$

Equation (2.3) shows that WFO is a nonlinear method. Therefore, it is more complicated to analyze because it depends on the parameter λ . However, if $\lambda = 0$, then WFO reduces to δidf . Table 2.3 summarizes the the linearity property of the three methods.

Table 2.3: Linearity of Supervised Term Weighting Methods.

Method	Relation	Category
rf	$p^+ = (\alpha - 2) \frac{N^-}{N^+} p^-$	Linear
δidf	$p^+ = \alpha p^-$	Linear
WFO	$p^- = \frac{p^+}{\beta p^{+\frac{\lambda}{\lambda-1}}}$	Nonlinear

From the above analysis, we can conclude that these methods only take into consideration the odds ratio of terms, which is based on class term presence. They do not consider the overall term distribution across the entire dataset. Therefore, they do not perform well on datasets with highly imbalanced class term distributions.

2.2 Text Categorization

Text categorization automatically classifies a set of text documents into different pre-defined categories. A considerable amount of literature has been published on text categorization. In this section, we will review machine learning text categorization approaches, which have been combined successfully with information retrieval in many real world applications.

2.2.1 Machine Learning Classifiers

Since text categorization is a typical high dimensional and sparse problem, numerous studies have attempted to reduce the dimensionality before applying machine learning. The standard approach is to apply stop-word (words with little semantic meaning) removal. For Latin-based languages, stemming [32] can be applied to reduce words to their root forms, e.g., {stemming, stemmed, stems} are all reduced to the root form *stem*.

The measure of uncertainty in information theory can also be used for dimensionality reduction. Measures that have been applied successfully in text categorization include gain ratio [24], Chi square [86], and mutual information [55]. Several studies have revealed that classifiers perform more accurately in the reduced space compared to the original space.

The number of machine learning algorithms applied in text categorization is bewildering. These consist of probabilistic methods, decision tree, Naïve Bayes [52, 54], K-Nearest-Neighbor [19], C4.5 [76], Support Vector Machine (SVM) [18], etc. These algorithms have all been applied successfully for text categorization.

In [41, 42], Joachims first introduced SVM for text categorization. Later, SVM was applied to spam categorization [27, 60, 46], Web content classification [29], and sentiment analysis [74]. According to Joachims, SVM is robust to overfitting and can scale-up to very high dimensions. Previous studies have found SVM to be one of the most successful algorithms for text categorization.

Recently, Dey [25] proposed a classification model for each pre-defined class using a term frequency and document scoring technique. If a document is misclassified by one of these models, it will then be re-classified by other models. When applied to the BBC Sports corpus, the proposed method achieved a macro-averaged F-measure of 94.7%.

In [56], Li et al. used a corpus-based thesaurus and WordNet to improve text categorization performance. They conducted experiments on the Reuters-21578 and the 20-Newsgroups datasets. Their proposed methods achieved improvements in all three measures of precision, recall, and F-measure.

Recent studies have also suggested that SVM is actually the best algorithm for sentiment analysis and binary text categorization problems compared to other machine learning algorithms [74]. Martineau et al. [63] also confirmed that SVM in association with supervised term weighting techniques performs significantly better than other methods for binary text categorization. The classification accuracy of SVM based on supervised term weighting can improve by more than 6% compared to other methods.

2.2.2 None Vector Space Model Approach

In addition to approaches based on the Vector Space Model mentioned in the previous section, there exist other dictionary-based approaches to classify documents. These approaches use Natural Language Processing techniques to build the dictionaries. In particular, they are often utilized in Sentiment Analysis, a special case of text categorization where the textual data is classified into two (positive or negative) or three categories (positive, negative, or neutral).

In [38, 39, 88], the authors first identify specific positive and negative terms. They then combine the sentiment contribution of each terms to calculate the overall sentiment of the text.

In [30], Esuli and Sebastiani built a lexical resource for opinion mining, which is a Wordnet synset associated with three scores, namely objective, positive, and negative. The sentiment of a document is determined by the average numerical scores of the constituent terms.

2.2.3 Centroid-based Classifiers

In [37], Han and Karypis analyzed and presented a linear-time centroid-based text classification algorithm, which outperformed a number of classical classifiers including Naïve Bayes [52, 54], K-Nearest-Neighbor [19], and C4.5 decision trees [76].

Building upon Han’s work, Guan et al. [35] subsequently proposed the Class-Feature-Centroid (CFC) classifier, which boosts the importance of highly discriminative terms. The CFC has been evaluated on multi-class textual datasets and shown to even beat the state-of-the-art SVM classifier. In their comparison, CFC and SVM were both configured to use the same feature space, namely $tf \times idf$ [62, 79], which is a popular unsupervised term weighting technique widely used in information retrieval.

2.2.4 Discussion

Notably, SVM is one of the best algorithms in text categorization but SVM in association with supervised term weighting methods performs better than SVM based on unsupervised term weighting methods. Hence, using a suitable supervised term weighting scheme, one can further improve the classification accuracy of even the best SVM classifier.

Moreover, compared to SVM and other machine learning algorithms, centroid-based classifiers are fast, efficient, and easy to deploy. Although CFC works very well in text categorization, it can be further improved by adjusting the importance of each term in a supervised

setting.

2.3 Perceptron-based Online Learning

2.3.1 Online Learning Problem Setting

Online learning operates on a sequence of data with time stamps. At time step t , the algorithm learns the weight $\mathbf{w}_t$ of the prediction function $f(\mathbf{x}_t) = \mathbf{w}_t \cdot \mathbf{x}_t$ by processing a sample $\mathbf{x}_t \in \mathbb{R}^n$ and predicting its label $\hat{y}_t \in \{-1, +1\}$. After prediction, it computes the loss $\ell(y_t, \hat{y}_t)$ which is the difference between its prediction and the revealed true label $y_t \in \{-1, +1\}$.

Online linear classification algorithms have been studied for close to 50 years, starting with the Perceptron [7, 70]. If the Perceptron algorithm makes a mistake, it will update the weight as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y_t \mathbf{x}_t$$

where η is a positive learning rate. Notably, it is difficult to choose a suitable learning rate.

There is also a related class of Bandit algorithms [47], whose learning process is similar to the Perceptron algorithm. However, in the prediction phase, the Bandit algorithm does not know the true label of the instance. The Bandit algorithm is actually more realistic with respect to real-world online tasks like micro-blog classification, since in practice the real class label is not known after each prediction unless the user constantly validates every prediction.

Other online learning algorithms use the Newton weight update method, including the LaRank and the OLaRank algorithms [10, 11]. Yet others are inspired by Support Vector Machines [18] and the Huller algorithm [9].

2.3.2 Online Passive Aggressive Learning

Recently, the large margin technique has been applied to Perceptron-based online learning algorithms. Preliminary work on large margin online learning algorithms, namely Passive Aggressive (PA) algorithms, was undertaken by Crammer et al. [21]. In online PA learning, the loss is used to update the weight with respect to some criteria. The goal is to achieve a margin of at least 1. Thus if the margin is less than 1 during a certain round, the algorithm suffers a loss. The loss can be modeled using the hinge-loss function, which equals zero when

the margin exceeds 1, as shown below:

$$\ell(\mathbf{w}; (\mathbf{x}, y)) = \begin{cases} 0 & y(\mathbf{w} \cdot \mathbf{x}) \geq 1 \\ 1 - y(\mathbf{w} \cdot \mathbf{x}) & \text{otherwise} \end{cases} \quad (2.4)$$

The overall objective of online learning is to minimize the cumulative loss over the entire sequence of examples. Crammer et al. [21] formulated it as an optimization problem and derived three versions of the PA algorithm. First, the optimization problem is formulated as follows:

$$\begin{aligned} \mathbf{w}_{t+1} = & \underset{\mathbf{w} \in \mathcal{R}^n}{\operatorname{argmin}} \quad \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 \\ & \text{such that} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0 \end{aligned} \quad (2.5)$$

Crammer et al. update the weight vector $\mathbf{w}_{t+1}$ at each round as

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t \quad \text{and} \quad \tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2} \quad (\text{PA}) \quad (2.6)$$

Second, to allow for incorrect predictions, a slack variable ξ was introduced into the optimization problem in Equation (2.5) with two penalties: linear and quadratic. The weight update equation to the soft-margin problem has the same form as that of (2.6), but with the weight coefficient τ_t defined as follows:

$$\tau_t = \min \left\{ C, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2} \right\} \quad (\text{PA-I}) \quad \text{and} \quad \tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} \quad (\text{PA-II}) \quad (2.7)$$

2.3.3 Distribution-aware Online Learning

Since 2005, there has been a renewed interest in Perceptron-based algorithms like the Second-order Perceptron (SOP) [15], which takes into account the data distribution. If the SOP algorithm makes a mistake, it will update the weight as follows:

$$\mathbf{w}_t = (aI_n + S_t S_t^T)^{-1} v_{k-1}$$

where $v_k = v_{k-1} + y_t \mathbf{x}_t$, a is a positive parameter, and S_t is a correlation matrix.

After the SOP algorithm has been proposed, another family of Perceptron-based algorithms, the Passive Aggressive (PA) algorithms [21] were introduced. The PA algorithms incorporate the margin maximizing criterion of modern machine learning algorithms. Although the PA algorithms perform better than the Perceptron algorithm, they do not take into account the distribution of data. As such, they could be improved further by considering the data distribution.

Algorithms that improved upon the PA algorithms by taking into account the distribution of data include the Confidence Weighted (CW) linear classification [26] and its latest version, the CW algorithm for multi-class classification [22]. CW assumes that the weight at each time step is Gaussian distributed with a mean vector and covariance matrix.

Using a probabilistic approach, the Confidence Weighted online learning algorithm learns a Gaussian distribution of weights with mean vector μ and covariance matrix Σ . The weight distribution is updated by minimizing the Kullback-Leibler divergence between the new weight distribution and the old one while ensuring that the probability of correct classification is greater than a threshold as follows:

$$\begin{aligned} (\mu_{t+1}, \Sigma_{t+1}) = & \underset{\mu, \Sigma}{\operatorname{argmin}} D_{KL}(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu_t, \Sigma_t)) \\ \text{such that} \quad & Pr_{\mathbf{w} \sim \mathcal{N}(\mu, \Sigma)}[y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 0] \geq \eta \end{aligned} \quad (2.8)$$

where Pr denotes the point probability. This optimization problem has a closed form solution:

$$\mu_{t+1} = \mu_t + \alpha_t y_t \Sigma_t \mathbf{x}_t \quad (2.9)$$

$$\Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t \mathbf{x}_t^T \mathbf{x}_t \Sigma_t \quad (2.10)$$

where $\alpha_t = \frac{-(1+2\phi M_t) + \sqrt{(1+2\phi M_t)^2 - 8\phi(M_t - \phi V_t)}}{4\phi V_t}$, $\beta_t = \frac{2\alpha_t \phi}{1+2\alpha_t \phi V_t}$, $V_t = \mathbf{x}_t^T \Sigma_t \mathbf{x}_t$, $M_t = y_t(\mu_t \cdot \mathbf{x}_t)$, and ϕ is a confidence parameter depending on η .

2.3.4 Two-view Online Learning

Assuming that we have two views of the same data $(\mathbf{x}_t^A, \mathbf{x}_t^B) \in \mathbb{R}^n \times \mathbb{R}^m$, where A and B stand for view A and view B respectively.

The PA algorithm works better than the SOP in terms of both speed and accuracy. However, it can only process one view at once. Whereas, in 1998, Blum and Mitchell proposed the co-training [8], which uses the unlabelled sample to boost the performance of a learning algorithm. In this problem setting, each training instance consists of two distinct views. The proposed algorithm was shown to work well on the Web page classification dataset. Later, Farquhar et al. [31] proposed a large margin two-view SVM algorithm called SVM-2K, which is an extension of the well-known SVM algorithm [18]. SVM-2k takes into account the correlation between the two views to improve classification accuracy. SVM-2k was shown to give better performance compared to the original SVM on different image datasets [31]. Similar to SVM, SVM-2k maintains two weight vectors $\mathbf{w}^A$ and $\mathbf{w}^B$, which are the solutions

of the following optimization problem:

$$(\mathbf{w}^A, \mathbf{w}^B, b^A, b^B) = \operatorname{argmin} \frac{1}{2} \|\mathbf{w}^A\|^2 + \frac{1}{2} \|\mathbf{w}^B\|^2 + C^A \sum_{i=1}^{\ell} \xi_i^A + C^B \sum_{i=1}^{\ell} \xi_i^B + D \sum_{i=1}^{\ell} \eta_i$$

$$\text{such that } |\langle \mathbf{w}^A, \mathbf{x}_i^A \rangle + b^A - \langle \mathbf{w}^B, \mathbf{x}_i^B \rangle - b^B| \leq \eta_i + \epsilon$$

$$y_i(\langle \mathbf{w}^A, \mathbf{x}_i^A \rangle + b^A) \geq 1 - \xi_i^A$$

$$y_i(\langle \mathbf{w}^B, \mathbf{x}_i^B \rangle + b^B) \geq 1 - \xi_i^B$$

$$\xi_i^A \geq 0, \xi_i^B \geq 0, \eta_i \geq 0, \text{ for all } 1 \leq i \leq \ell.$$

where the first constraint is used to penalize the disagreement between the two views.

2.3.5 Online Learning-to-Rank

For the past few years, many learning-to-rank algorithms have been proposed to improve the ranking of search results. Some were adapted from binary classification algorithms such as the Perceptron and its variants [7].

Suppose that R and N denote relevant and non-relevant feature score vectors with respect to a user query $\mathbf{q}$. Then the feature score vectors of relevant and non-relevant documents with respect to the query $\mathbf{q}$ are defined as ϕ_t^R and ϕ_t^N , respectively. Let $(\phi_t^R, \phi_t^N)_{t=1, T}$ be a sequence of training examples. We should have $\mathbf{w} \cdot \phi_t^R \geq \mathbf{w} \cdot \phi_t^N$ because the score of the relevant document to the query should be greater than that of the non-relevant document. In [33], Gao et al. applied the Perceptron algorithm to learn the ranking function, where the weight vector $\mathbf{w}_{t+1}$ is learnt on each round t as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau(\phi_t^R - \phi_t^N)$$

where τ is a constant learning rate.

This approach can improve the performance of search systems compared to classical information retrieval approaches. However, it shares the same limitations as the Perceptron algorithm; it performs poorly on datasets that are not linearly separable. Furthermore, the algorithm is very sensitive to the learning rate.

2.3.6 Discussion

The PA algorithm works better than the Perceptron algorithm. One question that needs to be asked, however, is whether distribution-aware strategies like SOP can boost the performance of the PA algorithm.

For learning-to-rank problems, in the realm of batch learning, the SVM algorithm [14, 90] has also been applied. Perhaps the most serious disadvantage of this approach is that for dynamic systems like math Q&A systems, where questions are posted every minute, an online learning algorithm is more desirable; it incrementally learns from the data as and when new data is added. Therefore, the problem now boils down to extending the PA algorithm to learn the scoring function weight.

2.4 Summary

In this chapter, we have surveyed the following topics: term weighting, text categorization, and online learning. These methods will be applied to text categorization, sentiment analysis, and information retrieval, where term weighting will be used in document preprocessing and online learning will be used in document classification. We also discussed the limitations of existing approaches and proposed ideas to tackle them.

Chapter 3

Kullback-Leibler Term Weighting

Recently, various machine learning and semantic orientation techniques have been proposed for binary text categorization. Pang et al. [74] applied supervised machine learning techniques to classify sentiments. Ku et al. [48] tried opinion extraction and summarization, while Hatzivassiloglou [38] used semantic orientation of adjectives. Among these, the Support Vector Machine (SVM) [18] approach based on the vector space model has been the most popular. In the vector space model, documents are first converted into a set of vectors, where each document is represented by a high-dimensional document vector with each dimension corresponding to a term weight. The choice of term weights during the conversion can affect subsequent classification performance. One of the most popular weighting schemes to date is $tf \times idf$ [62], which combines term frequency (tf) and inverse document frequency (idf). The $tf \times idf$ has been applied successfully to many classical information retrieval tasks and categorization problems. However, it does not make use of the difference in distribution of terms across classes.

In this chapter we propose a novel supervised term weighting approach, which addresses some of the limitations of supervised term weighting methods surveyed in Chapter 2. We then evaluate our proposed term weighting method on text categorization tasks, in particular sentiment analysis. Our proposed term weighting method called $tf \times KL$, which is shown to be superior to the class agnostic $tf \times idf$ for binary classification problems. Specifically, we express $tf \times KL$ and $tf \times idf$ as transformations under the SVM [18] classification framework.

3.1 Term Weighting Formulation

In the bag-of-words model, each document is represented by a bag of words/terms. Each term is weighted using both local and global information. The local component contains information specific to the individual document, while the global component is computed over the entire collection of documents. The term weighting can be formulated as follows:

$$w_{ij} = f(L_{ij}, G_j)$$

where L_{ij} is the local weight of term j in document i , G_j is the global weight of term j , and f is the term weighting function. In practice, terms can be weighted by term occurrence, term frequency, or $tf \times idf$, which is defined as

$$tf \times idf(x) = tf \times \log \frac{N}{n(x)}$$

where N is the total number of documents and $n(x)$ is the number of documents containing term x , i.e., document frequency of x . In this formula, tf and $idf = \log \frac{N}{n(x)}$ are the local and global weights of term x , respectively.

Lan's $tf \times rf$ method is a two class extension to $tf \times idf$, where the idf component is modified as follows:

$$tf \times rf(x) = tf \times \log_2 \left(2 + \frac{n^+(x)}{\max(1, n^-(x))} \right)$$

where $n^+(x)$ and $n^-(x)$ are the number of positive and negative class-labeled documents containing term x , respectively. Intuitively, $tf \times rf$ tries to give higher weightage to terms that appear more frequently in the positive class. However, there are some peculiarities with $tf \times rf$. First, $tf \times rf$ is asymmetric because it favors terms that appear more in the positive class only. This means that in practical two-class problems, the choice of the positive class will have an impact on term weighting and ultimately the classification performance. Second, $tf \times rf$ does not penalize terms that appear equally often in both classes. In both cases, terms are simply assigned their tf weights¹.

These observations have led us to consider other variations of supervised term weighting. Drawing on the $tf \times idf$ and $tf \times rf$ philosophy, a supervised term weighting method should have the following desirable properties:

- Boost the weight for term x if it occurs predominantly in many documents of one class only, be it positive or negative.

¹for uniformly distributed terms, those that appear equally in two classes, they are assigned a constant weight of $1.58tf$

- Attenuate the weight for term x if it occurs only in one class but over fewer number of documents.
- Reduce the weight for term x to zero if it occurs uniformly across both classes.

In the following subsections, we describe various impurity measures that can be used as alternative supervised term weights, and also derive our $tf \times KL$ term weighting method using the Kullback-Leibler divergence.

3.1.1 Impurity Measures

The three common class impurity measures, namely Entropy, Gini, and Classification Error (CE) [83], are defined as follows:

$$\begin{aligned} \text{Entropy}(x) &= - \sum_i^C p_i(x) \log_2(p_i(x)) \\ \text{Gini}(x) &= 1 - \sum_i^C p_i(x)^2 \\ \text{CE}(x) &= 1 - \max_{i=1,\dots,C} (p_i(x)) \end{aligned}$$

where $p_i(x)$ is the probability of term x belonging to class C_i . If term x occurs more frequently in one class only, its discriminative power is arguably higher. Hence, we define three new supervised term weighting scores as follows:

$$W_{\text{imp}}(x) = \begin{cases} \frac{1}{\text{imp}(x)} & \text{if } \text{imp}(x) \neq 0 \\ 1.0 & \text{otherwise} \end{cases} \quad (3.1)$$

where $\text{imp}(x) = \{\text{Ent}(x), \text{Gini}(x), \text{CE}(x)\}$.

We note that in the extreme case where x appears only in one class and not the other, all three impurity weights are theoretically infinite. Can we just set the weights of these terms to infinity? The practical answer is no, as it will lead to instability in the classifier. In practice, we have experimented with setting the extreme term weight to various large constants and found all of them to be unstable. We therefore bound the extreme weights to 1, which delegates all remaining weighting responsibility to the corresponding tf for the extremely discriminative terms.

Interestingly, the three impurity based term weights all share the same SVM kernel form as follows.

$$K(\mathbf{x}, \mathbf{x}_k) = \varphi(\mathbf{x}_k)\varphi(\mathbf{x}) = \mathbf{x}_k^T W_{\text{imp}}^T W_{\text{imp}} \mathbf{x}$$

3.1.2 Kullback-Leibler Divergence Measure

Divergence, also known as pseudo-distance, plays a crucial role in information theory and statistics. It expresses the likelihood ratios between pairs of multivariate probability distribution densities. We consider the Kullback-Leibler (KL) divergence, a special case of a broader class of divergences known as Ali-Silvey distances or f-divergence [1]. Intuitively, KL divergence measures the “distance” between two distributions p and q , which can be computed practically as

$$D_{\text{KL}}(p \parallel q) = \sum p \log\left(\frac{p}{q}\right)$$

Suppose that we have two point probabilities $p(x)$ and $q(x)$ so that $x \in C_1$ and $x \in C_2$, respectively. Let $h(x)$ denote the probability distribution of x over the corpus. The distances of term x to $p(x)$ and $q(x)$ are $D_{\text{KL}}(h \parallel p)$ and $D_{\text{KL}}(h \parallel q)$ respectively, such that

$$D_{\text{KL}}(h \parallel p) = \sum_x h(x) \log\left(\frac{h(x)}{p(x)}\right)$$

and

$$D_{\text{KL}}(h \parallel q) = \sum_x h(x) \log\left(\frac{h(x)}{q(x)}\right)$$

If the first merit is greater than the second, then x is closer to $p(x)$ than $q(x)$, and vice versa. Clearly, the role of term x in discrimination is directly proportional to the difference between the two merits. If the two merits are the same, the discriminatory role of term x should be small. Therefore, we define a new term weighting score for term x based on the KL divergence as follows:

$$W_{\text{KL}}(x) = |D_{\text{KL}}(h \parallel q) - D_{\text{KL}}(h \parallel p)| = |h(x) \log \frac{p(x)}{q(x)}|$$

This weight is equal to zero if the merits of term x to both classes are the same. Otherwise, the bigger the weight value, the more important term x is. Let $n^+(x)$ and $n^-(x)$ denote the number of documents containing term x in the positive and negative classes, respectively, we then have

$$p(x) = \frac{n^+(x)}{n(x)} \text{ and } q(x) = \frac{n^-(x)}{n(x)}$$

where $n(x)$ is the total number of documents containing term x . We have

$$W_{\text{KL}}(x) = |h(x) \log \frac{n^+(x)}{n^-(x)}| = h(x) \left| \log \frac{n^+(x)}{n^-(x)} \right|$$

where $h(x)$ can be approximated by $n(x)/N$, which is basically the normalized document frequency df . Hence, we have the final weight score as follows:

$$W_{KL}(x) \simeq \begin{cases} df \left| \log \frac{n^+(x)}{n^-(x)} \right| & \text{if } n^+(x) \neq 0 \text{ and} \\ & n^-(x) \neq 0 \\ 1.0 & \text{otherwise} \end{cases} \quad (3.2)$$

One nice property of W_{KL} is its inherent regulatory mechanism to deal with trivial or outlier proportions. Suppose a term “bush” appears in just 2 positive documents and 1 negative document of a training corpus with 1000 documents. Then, despite the two to one “discriminatory” power of the term “bush” ($n^+(\text{bush}) = 2$ and $n^-(\text{bush}) = 1$), the regulating normalized document frequency term $df = 3/1000$ will downplay the overall discriminatory effect of the word “bush”. More generally, $\log(n^+/n^-)$ remains constant when $n^+ = ka_1$ and $n^- = ka_2$ for some multiple k where a_1 and a_2 are constants, e.g., $n^+/n^- = 2/1 = 4/2 = 8/4 = \dots$ ($a_1 = 2$ and $a_2 = 1$). This is in stark contrast to $tf \times rf$, which regards a trivial proportion 2/1 as importantly as a more convincing proportion 2000/1000. Thus, the role of df in $tf \times KL$ is to boost the proportion $\log(n^+/n^-)$ for non-trivial terms, i.e., words that occur commonly enough (high normalized df). In other words, $tf \times KL$ is highly resistant to small differences in term class distributions. Table 3.2 list examples of some typical mixing proportions.

As with the three impurity based measures, the linear SVM kernel for $tf \times KL$ is defined as

$$K(\mathbf{x}, \mathbf{x}_k) = \varphi(\mathbf{x}_k)\varphi(\mathbf{x}) = \mathbf{x}_k^T W_{KL}^T W_{KL} \mathbf{x}$$

where W_{KL} is the KL weighting matrix.

Similar to $tf \times rf$, we combine the inverse impurity and KL weights with the tf value to define our new term weighting schemes as $tf \times Ent$ (Entropy), $tf \times GI$ (Gini), $tf \times CE$ (Misclassification Error), and $tf \times KL$. Table 3.1 summarizes the various supervised term weighting schemes.

3.1.3 Comparing $tf \times rf$ and $tf \times KL$

Among the term weighting methods, $tf \times rf$ and $tf \times KL$ are similar in the normal cases, but are totally different in the extreme cases. Therefore, to better understand their differences,

Table 3.1: Supervised Term Weighting Schemes.

Name	Weight Score
$tf \times idf$	$tf \times \log \frac{n}{n^+(x) + n^-(x)}$
$tf \times rf$	$tf \times \log_2 \left(2 + \frac{n^+(x)}{\max(1, n^-(x))} \right)$
$tf \times Ent$	$tf \times \frac{1}{-p(x) \log_2(p(x)) - q(x) \log_2(q(x))}$
$tf \times GI$	$tf \times \frac{1}{1 - p(x)^2 - q(x)^2}$
$tf \times CE$	$tf \times \frac{1}{1 - \max(p(x), q(x))}$
$tf \times KL$	$tf \times df \mid \log \frac{n^+(x)}{n^-(x)} \mid$

we compare them in Table 3.2 at various values of $n^+(x)$ and $n^-(x)$. We exclude other methods since they are totally different from these two. We highlight the similarities and differences of the two methods as follows:

1. Both give zero weights to terms not present in the training corpus.
2. When a term appears only in the negative class, $tf \times rf$ treats it just like any other term, assigning it a simple term frequency weight. On the contrary $tf \times KL$ will give higher weight to this term if it appears in a significant number of negative documents, as decided by the df value.
3. When a term appears only in the positive class, $tf \times rf$ gives this term preferential treatment, boosting it with a factor of $\log_2(2 + n^+(x))$ as shown in rows 3 and 4 of Table 3.2. Whereas $tf \times KL$ weighs it the same way if it appears only in either the positive or negative class. For $tf \times rf$, not all classes are equal, the positive class enjoys more privileges.
4. When a term appears much more frequently in the positive class compared to the negative class, $tf \times rf$ boosts its weight greatly. But again, as described before in the extreme cases, $tf \times rf$ discriminates against terms that appear much more frequently in the negative class, by asymptotically setting those term weights to tf . In contrast, $tf \times KL$ will boost the weight of a term if it has a skewed class distribution that is non-trivial (as regulated by the df term). Again, for $tf \times rf$, not all classes are equal, the positive class enjoys more privileges.
5. In the special case when a term is uniformly distributed across both positive and negative classes, $tf \times rf$ again delegates the weighting responsibility to tf , artificially

boosting it by a factor of 1.58, which is quite trivial in general. On the other hand, $tf \times KL$ penalizes terms that are uniformly distributed across two classes by setting their weights to zero.

Table 3.2: $tf \times rf$ vs. $tf \times KL$ (* denotes non-zero value).

n^+	n^-	$tf \times rf$	$tf \times KL$
0	0	0	0
0	*	tf	$tf \times df$
*	0	$tf \times \log_2(2 + n^+(x))$	$tf \times df$
*	*	$tf \times \log_2(2 + \frac{n^+(x)}{n^-(x)})$	$tf \times df \mid \log \frac{n^+(x)}{n^-(x)} \mid$
Special Cases (assuming N=1000)			
k	k	$1.58 tf$	0
2	1	$2 tf$	$0.0009 tf$
4	2	$2 tf$	$0.0018 tf$
400	200	$2 tf$	$0.18 tf$

In summary, there are fundamental differences between $tf \times rf$ and $tf \times KL$, mainly in the preferential way in which $tf \times rf$ treats a positive class, and also in how $tf \times rf$ ignores uniformly distributed terms. In general, $tf \times rf$ chooses to play safe by defaulting to the term frequency for most of the extreme cases, whereas $tf \times KL$ adopts a bolder term weighting philosophy that is less forgiving of skewed terms with high df .

3.1.4 Generalized $tf \times KL$

The $tf \times KL$ method directly depends on the ratio $\frac{n^+}{n^-}$, which is very sensitive to imbalanced data. Consider the example of a term “bush” appearing in 20 positive documents and 20 negative documents, but there are altogether only 20 positive documents and 980 negative documents. In this example, $tf \times KL$ will incorrectly penalize the term “bush” to have a weight of zero simply because it appears equally often (raw counts) in both classes! For the same example, $tf \times rf$ is more resistant; it will weigh “bush” by its default tf value. Therefore, $tf \times rf$ is more robust in handling imbalanced class distribution, which we have observed in practice.

As such, we propose a generalized version of $tf \times KL$ weighting to handle imbalanced datasets. Consider a term x and let N^+ and N^- be the number of positive and negative training documents. We generalize our proposed $tf \times KL$ method by replacing the term count with its class conditioned term probability as follows:

We have

$$P(x | +) = \frac{n^+(x)}{N^+} \frac{N}{n^+(x) + n^-(x)} \text{ and } P(x | -) = \frac{n^-(x)}{N^-} \frac{N}{n^+(x) + n^-(x)}$$

Replacing the term counts by the conditional probabilities, we have the generalized form of the KL term weights as follows:

$$W_{KL}(x) \simeq \begin{cases} df \mid \log \left(\frac{n^+(x)}{n^-(x)} \frac{N^-}{N^+} \right) \mid & \text{if } n^+(x) \neq 0 \text{ and } n^-(x) \neq 0 \\ 1.0 & \text{otherwise} \end{cases} \quad (3.3)$$

3.2 Performance Evaluation

In this section, we evaluate the SVM (using LibSVM [16]) classification performance of each term weighting model based on 4 benchmark datasets. We also study the impact of the amount of labeled data on the performance of each term weighting model, since labeled data is expensive to come by in practice. For performance comparison, we choose $tf \times idf$ and $tf \times rf$ as major competitors since they are currently the best methods [51].

3.2.1 Datasets

Four benchmark datasets were used, three binary sentiment datasets and one multi-class classification dataset. The details of these datasets are listed in Table 3.3. They are briefly discussed as follows:

Table 3.3: Datasets (* denotes multi-class).

Dataset	Terms	N	N^+
Movie Review	36,911	2,000	1,000
Sentiment Polarity	15,912	10,662	5,331
20 Newsgroups*	26,214	11,846	900
Multi-domain Sentiment	473,456	8,000	4,000

- **Movie Review.** The binary movie review dataset (version 2.0) from Pang and Lee [72] contains 1000 positive and 1000 negative average length movie reviews.
- **Sentiment Polarity.** The binary Sentiment Polarity dataset [73] contains 5331 positive and 5331 negative snippets of short sentences.
- **Multi-domain Sentiment.** The Multi-domain sentiment dataset (version 2.0) [6] contains 8,000 samples with 473,456 unigram and bigram features. This large dataset will showcase how well our weighting method performs on high dimensional and big datasets.

- 20-Newsgroup. The 20-Newsgroup dataset contains 26,214 average length documents spanning 20 categories. The number of documents in each class is balanced. We use this dataset to evaluate our proposed approach on multi-class problems.

All documents were processed without stemming or stop-word removal.

3.2.2 Performance Comparison

We tested the supervised term weighting method using a linear SVM on the four datasets. We converted each document into a document vector using different term weighting methods. We used the linear SVM with default parameters as it has been shown to be very effective for text classification, which involves high dimensions. We compared our approach against other methods using 5-fold cross validation. The performance of each term weighting scheme is given in Table 3.4.

Table 3.4: 5-fold Cross Validation Accuracy.

Dataset	Movie Review	Sent. Polarity	20 Newsgroups	Multi-domain Sent.
tf	82.90%	74.09%	97.30%	80.85%
$tf \times idf$	83.25%	71.88%	97.25%	83.35%
$tf \times rf$	83.50%	73.59%	95.65%	82.20%
$tf \times Ent$	84.55%	69.91%	95.30%	80.70%
$tf \times GI$	85.95%	70.80%	95.02%	81.30%
$tf \times CE$	88.15%	68.50%	95.30%	80.80%
$tf \times KL$	90.75%	75.33%	97.40%	89.35%

From the table we see that $tf \times KL$ outperforms all other weighting schemes. We have the following observations for each dataset.

Movie Review

Clearly, the polarized nature of movie reviews contributed to the success of $tf \times KL$ and to a lesser extent, the three impurity based term weighting schemes $tf \times CE$, $tf \times GI$, and $tf \times Ent$. The worst performer is tf .

Sentiment Polarity

Purity based measures like $tf \times Ent$ and $tf \times CE$ performed significantly worse than the other methods including the baseline $tf \times idf$. This is because each sample in this dataset is very short, containing only a few words, which means that the role of tf (how many times a word appears in the sentence) is more important. This is evidenced by the result that tf has

relatively good performance compared to all other methods. In fact, tf works much better even than the commonly used $tf \times idf$! For short sentences, noise effects are amplified, i.e., the presence or absence of even one noisy word may tip the balance. In such situations, $tf \times rf$ performed reasonably well, just slightly worse than tf . Moreover, $tf \times KL$'s inherent self-regulating document frequency term helped propel it to the number one position, confirming its noise-tolerant capability.

20-Newsgroups

In this dataset, term weighting methods such as $tf \times rf$, $tf \times Ent$, $tf \times GI$, and $tf \times CE$ performed significantly worse than the plain tf and $tf \times idf$ methods. Just as in the case of the Sentiment Polarity dataset, the term tf plays a very important role in this dataset due to the relatively short length of the articles and the multi-class nature of the dataset. Again, the self-regulating feature of $tf \times KL$ made it the best performer. Note that the accuracy here is dominated by the majority “negative” class, and any small deviation in accuracy is significant.

Multi-domain Sentiment

In this dataset, $tf \times GI$ outperformed all other models except $tf \times idf$, $tf \times rf$, and $tf \times KL$. Again, $tf \times KL$ performed the best and is 6% better than the runner-up, which is the $tf \times idf$ baseline.

3.2.3 When to Use $tf \times KL$?

In this section, we study the effect of labeled data quantity on the performance of $tf \times KL$. To do this, we split the Multi-domain Sentiment dataset into 10 parts and calculate the $tf \times idf$ weights for the whole dataset as the baseline. We then calculate the term weighting matrix KL in Section 3.1.2 based on these parts. To compare the performance of $tf \times KL$ and $tf \times idf$, we use a linear SVM classifier. The experimental results are shown in Figure 3.1 where the dashed lines represent baseline $tf \times idf$ results for each of the 4 classes. The point where the solid lines crosses the dashed line of the same color gives the corresponding amount of labeled training data that is necessary for $tf \times KL$ to outperform plain old $tf \times idf$.

Figure 3.1: The Effect of Labeled Data based across Domains.

We see that, in the worst case (where the crossover occurs the latest, for the Kitchen domain), $tf \times KL$ is worse than $tf \times idf$ when the percentage of labeled data is less than 62%.

3.2.4 Performance on Imbalanced Datasets

We now pit the generalized $tf \times KL$ against $tf \times rf$ on highly skewed datasets, which are very common in real world sentiment data. We applied stratified sampling without replacement on the multi-domain sentiment dataset to obtain 9 imbalanced datasets, each with increasing proportion of positive training samples, i.e., 10%, 20%, ..., 90%. Each imbalanced dataset was fed into the linear SVM classifier using the various term-weighting methods, and the cross-validated F1, precision, and recall for both the positive and negative classes were computed.

Positive Class

The F1, Precision, and Recall scores along with their error bars for the positive class are shown in Figures 3.2 (a), (c), and (e), respectively. We see that the generalized $tf \times KL$ and other impurity methods consistently outperformed the $tf \times rf$ method. For example, if the

Figure 3.2: F1, Precision, Recall for Positive (+) and Negative (-) Classes on Imbalanced Dataset.

percentage of positive examples is 10%, the positive F1 measure is improved by up to 20% for $tf \times KL$. Even in the worst case, the F1 measure is improved by 3%. Figure 3.3 summarizes the gain of $tf \times KL$ over the second-place method, which can vary.

Figure 3.3: Improvement of $tf \times KL$ over Runner-up method for Positive Class.

Negative Class

Figure 3.2 (b), (d), and (f) shows the F1, precision, and recall scores for decreasing amount of negative labeled data, respectively. Note that the proportion of negative class training samples actually decreases from left to right, i.e., 10% positive (implies 90% negative), 20% positive (80% negative), etc. We see that the generalized $tf \times KL$ still leads for all proportions except one, the extreme skewed case of 90% positive (10% negative), at which point $tf \times CE$ is the leader in terms of F1. In fact, for this extreme case, the other two impurity based measures $tf \times Ent$ and $tf \times GI$ also performed similarly as the generalized $tf \times KL$. We also note the $tf \times rf$ gave the best negative class recall over the range for the 20%-50% negative proportions (50% to 80% positive in Figure 3.2(f)). The lost ground is regained in the higher precision of $tf \times KL$, resulting in overall best F1 score for $tf \times KL$.

Figure 3.4 summarizes the gain of $tf \times KL$ over the next best performer. Note that $tf \times KL$ lost in F1 from 80% onwards, and in recall from 50% onwards.

Figure 3.4: Improvement of $tf \times KL$ over Runner-up method for Negative Class.

3.3 Summary

In this chapter, we proposed a fundamentally sound supervised term weighting method, $tf \times KL$, which is derived from the Kullback-Liebler divergence. We systematically compared $tf \times KL$ with the recently proposed supervised term weighting approach $tf \times rf$ and showed why $tf \times KL$ is more well thought out, by examining the typical and extreme class distribution scenarios (using the generalized $tf \times KL$). In summary, $tf \times KL$ treats both positive and negative classes equally, is more strict about admitting skewed term distributions, and heavily penalizes uniformly distributed terms and trivial differences in class distributions.

Experimentally, we benchmarked $tf \times KL$ against 4 other supervised term weighting schemes along with the $tf \times idf$ baseline. The $tf \times KL$ was found to consistently lead the pack, gaining as much as 6% advantage against the runner-up on some datasets. For

imbalanced datasets, the generalized $tf \times KL$ performed up to 20% better than the runner-up in terms of positive class F1 measure, and continues to improve the performance even when the positive class becomes the decisive majority. For highly imbalanced negative class, $tf \times KL$ was also able to maintain its leading position in most cases.

Chapter 4

Centroid-based Classifiers

In Chapter 3, we have proposed a supervised term weighting approach, which was evaluated based on the batch learning SVM algorithm. In this chapter, we applied supervised term weighting to another class of classifiers, the centroid based classifier. The centroid based classifier is essentially an online linear learning algorithm, which can work with large-scale datasets. The centroid based classifier is simple and works as follows. During the training or learning phase, a centroid is computed for each class. During prediction, a test sample is classified to the class of the nearest centroid, where the “distance” function is commonly the dot product of the prototype and the testing example. We evaluated our proposed approach on both text categorization and sentiment analysis datasets.

Guan et al. [35] recently proposed a Class-Feature-Centroid (CFC) classifier that uses intra-class and inter-class term presence/absence to boost the weight of highly discriminative terms during training. Each term is weighted by its document frequency (the intra class information) and a class discriminative factor that is inversely proportional to the number of classes containing it (the inter-class information). In addition, CFC goes against conventional wisdom by 1) not normalizing the centroid during prediction, and 2) boosting the weights of words exclusive to a larger class. CFC was evaluated on the Reuters-21578 and 20-Newsgroup datasets and shown to significantly outperform the simple Arithmetical Average Centroid (AAC) and Cumuli Geometric Centroid (CGC) approaches as well as the well-known baseline SVM classifier. Moreover, CFC training is orders of magnitude faster than SVM, and it works well on multi-class datasets.

While the CFC has been shown to work very well on multi-class datasets in general, it suffers from two major drawbacks: 1) it performs very poorly for binary class data due to the aggressiveness in which it penalizes non-discriminatory terms, i.e., words that appear in both

classes, regardless of the actual counts; and 2) it is inherently biased towards the majority class, i.e., it performs poorly for highly imbalanced datasets.

To illustrate, consider a two-class dataset (positive and negative classes) where a term appears in both positive and negative documents. In CFC, this term will be assigned zero weight by virtue of its inter-class commonality. Essentially, what CFC is doing is a very aggressive form of feature selection; a term is useful if and only if it appears in as few numbers of classes as possible. One consequence of this aggressive weighting philosophy is that outlier terms like people names can become overly important, as will be shown later.

The second limitation of CFC is that it does not work well on imbalanced datasets, which we shall prove theoretically. For prediction, CFC uses the de-normalized cosine similarity function to find the nearest centroid to an unlabeled document. This essentially means that highly discriminative terms, i.e., those with exclusive class document frequency, will dominate the similarity comparison. A natural effect of this bias is that larger classes will have larger weights.

Borrowing the moment generating function from multi-set theory and applying it to the class document frequency (number of documents containing the term within the class), we show that the CFC centroid grows exponentially with respect to the class document frequency, while CGC and AGC centroids grow linearly. We subsequently propose an effective way to regulate the exponential CFC centroid growth by multiplying a discriminatory factor based on class-conditioned term probabilities.

4.1 Centroid-based Classification

Given a set of training document vectors $\{\mathbf{x}_n : n = 1, \dots, N; \mathbf{x}_n \in \mathbb{R}^D\}$ and the corresponding class labels $\{y_n : n = 1, \dots, N; y_n \in \{1, \dots, M\}\}$, the goal during the training phase is to determine M centroids, $\{\mathbf{c}_m : m = 1, \dots, M; \mathbf{c}_m \in \mathbb{R}^D\}$ corresponding to each of the M classes. The centroid $\mathbf{c}_m$ is also known as the *prototype* or *code book* vector for class C_m . Let $C_m = \{\mathbf{x}_n \mid y_n = m\}$ denote the set of all vectors from class m .

For prediction, an unlabeled document $\mathbf{x}$ is assigned to the class of its closest centroid, based on a similarity measure $\text{sim} : \{(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i \in \mathbb{R}^D, \mathbf{x}_j \in \mathbb{R}^D\} \rightarrow \{s \in \mathbb{R} \mid 0 \leq s \leq 1\}$

$$\hat{y}(\mathbf{x}) = \arg \max_{i=1, \dots, M} \text{sim}(\mathbf{x}, \mathbf{c}_i) \quad (4.1)$$

The most common similarity measure used for text classification is the cosine similarity [62].

Many methods have been proposed to compute the class prototype vectors, including classical Arithmetical Average Centroid (AAC) and Cumuli Geometric Centroid (CGC) [37]. In these approaches, the prototype vectors are the arithmetical average or cumulative sum of all documents within a class. For instance, the AAC and CGC approaches calculate the centroids as

$$\mathbf{c}_m^{\text{AAC}} = \frac{1}{|C_m|} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n \quad (4.2)$$

and

$$\mathbf{c}_m^{\text{CGC}} = \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n \quad (4.3)$$

respectively, where $m = 1, \dots, M$.

In CFC [35], the d -th entry of the m -th prototype vector $\mathbf{c}_m^{\text{CFC}} = [c_{m,1}^{\text{CFC}}, \dots, c_{m,D}^{\text{CFC}}]$ is computed as

$$c_{m,d}^{\text{CFC}} = b^{\frac{N^m(x_d)}{|C_m|}} \log \left(\frac{M}{\zeta(x_d)} \right) \quad (4.4)$$

where $N^m(x_d)$ is the document frequency of term x_d in class C_m and $\zeta(x_d)$ is the number of classes containing term x_d , and M is the total number of classes.

Inter-class information can certainly boost the importance of highly discriminative terms. However, assigning zero weight to a term that occurs in all classes, as dictated by CFC in the second part of Equation (4.4), is quite drastic because the same term may appear at different frequencies in different classes. To illustrate this problem, consider the following example.

Suppose we have a two-class dataset with class labels C_1 and C_2 . Each class contains two terms x_1 and x_2 , and two documents $C_1 = \{(20, 1), (80, 0)\}$ and $C_2 = \{(0, 10), (1, 90)\}$. Here, each document is represented as a term frequency row vector, e.g., the first document in class 1 contains 20 occurrences of term 1 (t_1) and one occurrence of term 2 (t_2). Table 4.1 shows the distribution of terms and CFC centroids. Clearly, term t_1 is unique to class C_1 while term t_2 is unique to class C_2 , but both terms will have zero weights in the CFC centroid! This happened because both terms occur (although in disproportionate frequencies) in each class!

Table 4.1: Term Counts (C_1, C_2) and Centroids ($\mathbf{c}_1, \mathbf{c}_2$) for an Extreme Example

Feature	C_1	C_2	$\mathbf{c}_1$	$\mathbf{c}_2$
t_1	100	1	0	0
t_2	1	100	0	0

Figure 4.1 shows the word clouds¹ of positive and negative classes in the Movie Review dataset [72] (details in Table 4.3). Common nouns like “movie”, “film”, “story” clearly dominate both classes. In addition, polarity words such as “like”, “good”, and “funny” appear with the same frequency in both classes! On the other hand, only a few words like “best” and “bad” appear more frequently in the positive and negative classes, respectively. This shows that the use of raw unweighted polarity words alone may not be sufficient, which led us to investigate how the weights of highly discriminatory words can be boosted.

Figure 4.1: Word Clouds

¹generated by Wordle (<http://www.wordle.net>)

In this section we present a word cloud model for text categorization, borrowing from multi-set theory. In this model, each class is represented by a multi-set of words, also known as a word cloud. We then classify an unlabeled document into the category of the closest class. The proximity is based on a “distance” metric of the document to each class. Without loss of generality, we assume that each word in a document can be represented by an integer, and formalize the problem as follows:

Definition 4.1 (Word Set). A *Word Set* A of words $t_1, t_2, \dots, t_n$ appearing in a corpus of documents is defined as

$$A = \{t_1, t_2, \dots, t_n\}$$

where $t_i \in \mathbb{N}^+$. Note that $A \subset \mathbb{N}^+$.

Definition 4.2 (Word Cloud). A *Word Cloud* D of a set of documents is defined as

$$D = (A, \mu)$$

where $\mu : A \mapsto \mathbb{N}^+$ is a vector member function, and each $\mu(t_i)$ indicates the number of times $t_i \in A$ appears in D .

Definition 4.3 (Moment Generating Function). A moment generating function $m : \mathbb{R}^A \mapsto \mathbb{R}^A$ of a word cloud $D = (A, \mu)$ is defined as

$$m_D(\mathbf{t}) = \mathbb{E}[e^{\mathbf{t}^T \mu(A)}]$$

where $\mathbb{E}[\cdot]$ is the expectation (average) operator. Note that the moment generating function is unique for each word cloud. Taking advantage of its uniqueness property, we are able to use it to determine the class of documents by comparing the “distance” between document word cloud and class word cloud.

Theorem 4.1 (Union). If $D_1, D_2, \dots, D_n$ are word clouds with the corresponding moment generating functions $m_{D_i} = \mathbb{E}[e^{\mathbf{t}^T \mu(A_i)}]$, then the moment generating function of the union $D = \bigcup_i D_i$ is defined as

$$m_D(\mathbf{t}) = \prod_i m_{D_i}(\mathbf{t})$$

Proof. Suppose that $D = (A, \mu(A))$, we have $A = \bigcup_i A_i$ and $\mu(D) = \sum_i \mu(A_i)$. Therefore, we have the moment generating function of D as follows:

$$m_D(\mathbf{t}) = \mathbb{E}[e^{\mathbf{t}^T \mu(A)}] = \mathbb{E}[e^{\sum_i \mathbf{t}^T \mu(A_i)}] = \prod_i \mathbb{E}[e^{\mathbf{t}^T \mu(A_i)}]$$

which concludes the proof. □

Corollary 4.1 (Mean Word Cloud). If $D_1, D_2, \dots, D_n$ are word clouds with the corresponding moment generating functions $m_{D_i} = \mathbb{E}[e^{\mathbf{t}^T \mu(A_i)}]$, then we have their mean word cloud defined as

$$\hat{D} = (\bigcup_i A_i, \frac{1}{n} \sum_i \mu(A_i))$$

and its moment generating function defined as

$$m_{\hat{D}} = \mathbb{E}[e^{\frac{1}{n} \sum_i \mu(A_i)}]$$

Definition 4.4 (Word Cloud Similarity). Let D_1 and D_2 be two word clouds, whose similarity is defined as the cosine similarity between their moment generating functions as follows:

$$\text{sim}(D_1, D_2) = \frac{\langle m_{D_1}(\mathbf{t}), m_{D_2}(\mathbf{t}) \rangle}{\| m_{D_1}(\mathbf{t}) \| \| m_{D_2}(\mathbf{t}) \|}$$

where $\langle \cdot \rangle$ is the dot product. It can be trivially shown that this similarity measure is a metric.

To use word cloud classification, we take the following steps:

- *Document Representation.* Convert documents into word clouds and determine their moment generating functions.
- *Centroid Computation.* Determine the moment generating function of each class from the moment generating functions of documents in the class.
- *Class Prediction.* Calculate the similarity between document word clouds and class word clouds based on their moment generating functions.

4.1.2 Centroid-based Classifier with Kullback-Leibler Divergence

In this section, we propose a centroid-based classifier called CFC-KL using the word cloud model.

Document Representation

Similar to the bag-of-words vector space model [62], we first preprocess documents by converting them into word clouds. Suppose A is a set of all terms in the corpus, each word cloud can be represented by a member function $\mu(A)$. Clearly, $\mu(A)$ is a term frequency vector in this space. Therefore, we are able to apply supervised term weighting technique $tf \times KL$ as mentioned in Chapter 3 to boost the performance of classifiers.

Learning the Centroids

In this section, we propose a new approach to determine the mean word cloud of classes based on Corollary 4.1. Recall that we have a set of documents $\{\mathbf{x}_n : \mathbf{x}_n \in \mathbb{R}^D; n = 1, \dots, N\}$. Suppose that $\mathbf{x}_n$ is also a term frequency vector. We have the moment generating function of the mean word cloud as follows:

$$m_{C_m} = \mathbb{E}[e^{\frac{1}{|C_m|} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n}] \quad (4.5)$$

Similar to $tf \times KL$ term weighting, we use the KL term weight to boost the classification accuracy by determining the centroid of class C_m as follows:

$$c_{KL}^m = e^{\frac{1}{|C_m|} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n} \times \bar{N}(\mathbf{x}_n) \sum_{k=1}^{M-1} \sum_{l=k+1}^M \left| \log \left(\frac{n^k(\mathbf{x}_n)}{n^l(\mathbf{x}_n)} \frac{N^l}{N^k} \right) \right| \quad (4.6)$$

where M is the number of classes.

In binary classification problems, the above formula can be simplified to

$$\mathbf{c}_{KL}^+ = e^{\frac{1}{N^+} \sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n} \times \bar{N}(\mathbf{x}_n) \left| \log \left(\frac{n^+(\mathbf{x}_n)N^-}{n^-(\mathbf{x}_n)N^+} \right) \right|$$

and

$$\mathbf{c}_{KL}^- = e^{\frac{1}{N^-} \sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n} \times \bar{N}(\mathbf{x}_n) \left| \log \left(\frac{n^+(\mathbf{x}_n)N^-}{n^-(\mathbf{x}_n)N^+} \right) \right|.$$

Prediction

During prediction, an unlabeled document is assigned the $tf \times KL$ term weights as presented in Chapter 3. We then compute the similarity between the unlabeled vector and all centroids, using the similarity function defined in Definition 4.4. Accordingly, the unlabeled document is classified to the most similar class as described in Equation (4.1).

4.1.3 Growth of Class Document Frequency

We now examine the rate of change of each term in centroid $\mathbf{c}$ with respect to a change in the class term frequency. We consider each term separately, assuming independence of the terms.

Proposition 4.1. Let t denote the class specific document frequency of a term for a class C with centroid $\mathbf{c}$. Then, each term $\{c_d \mid d = 1, \dots, D\}$ in the AAC or CGC centroid has a linear growth rate with respect to its class specific document frequency t , i.e., $dc/dt = -k/t$ where k is a constant that differs for each term.

Proof. Suppose each document in class C is weighted by $tf \times idf$, substituting $tf \times idf$ into the AAC or CGC centroid function for a particular term x , we have

$$\mathbf{c} = \frac{1}{|C|} \sum_{\mathbf{x}_n \in C} f(x_n) \log \frac{N}{t} \quad (4.7)$$

$$= \frac{1}{|C|} \log \frac{N}{t} \sum_{\mathbf{x}_n \in C} f(x_n) \quad (4.8)$$

$$= \frac{L}{|C|} \log \frac{N}{t} \quad (4.9)$$

where the class document frequency $N(x)$ is replaced by t and the local weights $f(x_n)$ are collected into the constant $L = \sum_{\mathbf{x}_n \in C} f(x_n)$. Taking the derivative of Equation (4.9) with respect to t , we obtain our result,

$$\frac{dc}{dt} = -\frac{L}{|C|t} = -\frac{k}{t} \quad (4.10)$$

where k is a constant for term x . The proof for CGC is trivially similar and therefore omitted. It can be easily shown that CGC has the same inverse growth rate with constant $k = L$. $\square$

Proposition 4.2. Each term $\{c_d \mid d = 1, \dots, D\}$ in the CFC centroid has an exponential growth rate with respect to its class specific document frequency t , i.e., $dc/dt = k_1 b^{k_2 t}$ where b is a base exponent, k_1 and k_2 are constants for each term.

Proof. First we rewrite Equation (4.4) in terms of the class document frequency t as follows:

$$c = b^{\frac{t}{|C|}} \log \left(\frac{M}{\zeta(x_d)} \right) \quad (4.11)$$

Taking the derivative with respect to t yields our result:

$$\frac{dc}{dt} = \frac{1}{|C|} \log b \log \left(\frac{M}{\zeta(x_d)} \right) b^{\frac{t}{|C|}} = k_1 b^{k_2 t} \quad (4.12)$$

where $k_1 = \log b \log [M/\zeta(x_d)] / |C|$ and $k_2 = 1/|C|$. $\square$

What Equation (4.12) tells us is that for every new document added to the centroid containing term x , causing its class document frequency t to increase by 1, the CFC centroid value for that term will increase exponentially by $k_1 b^{k_2}$. On the contrary, the AAC or CGC centroid will only increase by k . In the long run, the CFC centroid will be dominated completely by terms that are exclusive to the class based on their exponential growth. This also means that smaller classes will suffer greatly due to their exponentially slower growth (fewer class documents), thereby allowing the large classes to dominate the prediction. Compared to the CFC method, our proposed CFC-KL approach takes the form of an e-folding time scale function; it has the same property as the CFC but is simpler because the model parameter b need not be specified. Table 4.2 summarizes the growth rate of a centroid term with respect to the class document frequency for the various centroid based classifiers.

Table 4.2: Centroid Growth versus Class Document Frequency

	dc/dt	k_1	k_2
AAC	$-k_1/t$	$\frac{1}{ C } \sum_{\mathbf{x}_n \in C} f(x_n)$	-
CGC	$-k_1/t$	$\sum_{\mathbf{x}_n \in C} f(x_n)$	-
CFC	$k_1 b^{k_2 t}$	$\frac{1}{ C } \log b \log [M/\zeta(x_d)]$	$\frac{1}{ C }$
CFC-KL	$k_1 e^{k_2 t}$	$\bar{N}(x_d) \sum_i \sum_j \left \log \left(\frac{N^i(x_d)+1}{N^j(x_d)+1} \frac{N^j}{N^i} \right) \right $	$\frac{1}{ C }$

4.1.4 Centroid-based Classifier with Jensen-Shannon Divergence

The KL divergence measures the “distance” between two probability distributions. It has been used quite extensively in engineering and statistics, apart from our use of it to compute term weights. However, it has a few well-known limitations. First, it is not a true metric because it is not symmetric. Second, it does not follow the triangle inequality. Third, although the KL divergence is a non-negative function, it is unbounded. For example, in the extreme cases when $n^k(\mathbf{x}_n) = 0$ or $n^l(\mathbf{x}_n) = 0$ in Equation (4.6), the KL supervised term weight can approach infinity. Although smoothing techniques can be applied, the smoothed term weights may become intractable. These limitations led us to consider the more general Jensen-Shannon divergence for dealing with multiple classes of documents. Formally, the

Jensen-Shannon divergence for two probability distributions p_1 and p_2 is defined as

$$JS(p_1, p_2) = -(\pi_1 p_1 + \pi_2 p_2) \log(\pi_1 p_1 + \pi_2 p_2) + \pi_1 p_1 \log(p_1) + \pi_2 p_2 \log(p_2) \quad (4.13)$$

where $\pi_1 + \pi_2 = 1$, $\pi_1 \geq 0$, and $\pi_2 \geq 0$.

For M classes, the Jensen-Shannon (JS) divergence is defined more generally as

$$JS(p_1, \dots, p_M) = -\left(\sum_{i=1}^M \pi_i p_i\right) \log\left(\sum_{i=1}^M \pi_i p_i\right) + \sum_{i=1}^M \pi_i p_i \log(p_i) \quad (4.14)$$

where $\sum_{i=1}^M \pi_i = 1$ and $\pi_i \geq 0$ for all i .

Remarks. In the extreme case of $p_i = 0$ for all i , $p_i \log(p_i) = 0$ always hold because $\lim_{p_i \rightarrow 0} p_i \log(p_i) = 0$. Thus, in contrast to KL divergence, JS divergence works well in the extreme case and is symmetric. Moreover, according to Lin [59], JS divergence is bounded in $[0, 1]$. Compared to KL divergence, JS divergence uses a set of weights $\{\pi_i\}$ for the probability distributions $\{p_i\}$. By taking these weights into account, we are able to propose another term weighting approach that works well on multi-class imbalanced datasets.

In this section, a new term weighting approach based on Jensen-Shannon divergence is formulated as follows:

Document Representation

For the sake of simplicity, the JS term weighting approach is formulated for binary problems first. Similar to the KL term weighting approach $tf \times KL$, the supervised term weight KL is replaced by JS , which is defined as follows. Let $p^+ = n^+(x)/N^+$ and $p^- = n^-(x)/N^-$ be the probability of term x belonging to the positive and negative classes respectively.

$$JS(p^+, p^-) = -(\pi^+ p^+ + \pi^- p^-) \log(\pi^+ p^+ + \pi^- p^-) + \pi^+ p^+ \log(p^+) + \pi^- p^- \log(p^-) \quad (4.15)$$

where $\pi^+ + \pi^- = 1$, $\pi^+ \geq 0$, and $\pi^- \geq 0$. This new term weighting approach is named $tf \times JS$.

Remarks. If the dataset is balanced (uniform class distribution), then π^+ and π^- is set to 0.5. Otherwise, they assume values that is inversely proportional to their class probabilities. Specifically, we have $\pi^+ = N^-/N$ and $\pi^- = N^+/N$. In other words, if the number of examples in the positive class is larger than that of the negative class ($N^+ > N^-$), then π^- should be larger than π^+ and vice versa.

Learning the Centroids

Similar to $tf \times KL$, to compute the $tf \times JS$ centroid, the KL term weight is replaced by the JS term weight as

$$c_{JS}^m = e^{\frac{1}{|C_m|} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n} \times JS(p_1, \dots, p_M) \quad (4.16)$$

where M is the number of classes.

To calculate the JS term weights, the weights of the prior class probability distributions must be known in advance. Let N_i be the number of documents in class i , the weight π_i is determined based on the *softmax* function [64] as

$$\pi_i = \frac{e^{\frac{-N_i}{N}}}{\sum_{j=1}^n e^{\frac{-N_j}{N}}} \quad (4.17)$$

where N is the total number of documents and n is the number of classes. Clearly, π_i meets the conditions $\pi_i > 0$ and $\sum_i \pi_i = 1$.

Discussion. The JS divergence is bounded in $[0, 1]$ [59] and therefore the JS term weights are always non-negative and less than 1.0. Therefore, the term frequency weight could overshadow the JS term weights, rendering it ineffective. To balance the term frequency and JS term weights, a re-normalization process should be performed as follows.

Let $\bar{\mathbf{x}}_m = \frac{1}{|C_m|} \sum_{\mathbf{x}_n \in C_m} \mathbf{x}_n$ be the mean vector for class m , the normalized centroid should have the following form:

$$c_{JS}^m = e^{\frac{\bar{\mathbf{x}}_m}{\|\bar{\mathbf{x}}_m\|}} \times JS(p_1, \dots, p_M) \quad (4.18)$$

For the binary problem, we have

$$\mathbf{c}_{JS}^+ = e^{\frac{\bar{\mathbf{x}}^+}{\|\bar{\mathbf{x}}^+\|}} \times JS(p^+, p^-)$$

and

$$\mathbf{c}_{JS}^- = e^{\frac{\bar{\mathbf{x}}^-}{\|\bar{\mathbf{x}}^-\|}} \times JS(p^+, p^-)$$

Prediction

In the prediction phase, we use the cosine similarity function to calculate the distance between the document vectors and the centroids. With the cosine similarity function, we can correct

the bias for long documents. Also, we can revise the centroid calculation by using the normalized vector instead of the average vector. Therefore, $\frac{1}{N^+} \sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n$ and $\frac{1}{N^-} \sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n$ are replaced by $\frac{\sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n}{\|\sum_{\mathbf{x}_n \in C^+} \mathbf{x}_n\|}$ and $\frac{\sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n}{\|\sum_{\mathbf{x}_n \in C^-} \mathbf{x}_n\|}$, respectively. The new CFC algorithm based on the $tf \times JS$ term weighting approach is called CFC-JS.

4.2 Performance Evaluation

We compare the performance of our proposed CFC-KL and CFC-JS with the baseline CFC centroid classifier. In addition, we also evaluate the support vector machine (SVM) [43]² classifier weighted by $tf \times idf$ (called SVM-IDF), $tf \times rf$ (called SVM-RF), $tf \times KL$ (called SVM-KL), and $tf \times JS$ (called SVM-JS).

4.2.1 Datasets

We evaluate our proposed approaches on three binary datasets (movie review, sentiment polarity and multi-domain sentiment) and two multi-class datasets (Reuters-21578 and 20-newsgroups), as shown in Table 4.3. More details of the dataset can be found in Section 3.2.1 of Chapter 3. The Reuters-21578 dataset contains 8,293 documents belonging to 65 categories. Each document has about 8,293 terms. The number of documents in each category is not balanced. While the first category has 3,713 documents, the others just have a few documents, e.g., one or two documents per class.

Table 4.3: Datasets (* denotes multi-class)

Dataset	Terms	N	N^+
Movie Review	36,911	2,000	1,000
Sentiment Polarity	15,912	10,662	5,331
Multi-domain Sentiment	473,456	8,000	4,000
20-Newsgroup*	26,214	-	-
Reuters-21578*	18,933	-	-

4.2.2 Term Weighting Visualization

Figures 4.2, 4.3, 4.4, and 4.5 show the word clouds of four global term weighting methods idf , rf , KL , and JS , respectively. Here, the colors of terms correspond to the classes; negative, positive, and neutral (appearing in both classes) terms are shown in red, blue, and green, respectively. The terms are sized proportionally to their term weights. Their horizontal

²we use the LibSVM (<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>) library with linear kernel and other default parameters

Figure 4.4: KL Term Weight Visualization of the Movie Review Dataset

Figure 4.5: JS Term Weight Visualization of the Movie Review Dataset

Figure 4.4 shows the KL term weight word cloud. It assigns larger weights to both negative and positive terms, as shown by their larger font sizes at the edges. Neutral terms in the middle tend to have smaller weights. Take the highly negative word “horrid” as an example, which appears on the leftmost part (second column from left) of the word cloud with a relatively large font size.

The JS term weight word cloud is shown in Figure 4.5. The JS term weight word cloud is similar to the KL term weight word cloud. JS also assigns smaller weights for neutral

terms, who show up in the middle area. However, the difference between the term weights of sentiment and neutral terms are less pronounced, as shown by the similar font sizes of all words. This is due to JS weight been bounded to lie between 0 and 1.

4.2.3 Binary Classification

We evaluate our proposed KL and JS weighting approaches on binary text datasets, plotting their cross-validated F-measure, precision, and recall values along with error-bars denoting standard deviation.

The results for the movie review dataset are shown in Figure 4.6. We see that CFC-JS has the best F-measure, followed by CFC-KL in a close second place, leaving CFC in the dust (CFC-JS has 20% better F-measure compared to the baseline CFC).

When it comes to SVM based classifiers, the plain vanilla SVM-IDF beats the baseline CFC (we were unsuccessful in replicating the superiority of CFC over SVM as mentioned in [35]). CFC-JS managed to be better or similar to SVM-IDF in terms of F-measure, winning on recall but losing out a little on precision. Moreover, the overall best performer in terms of F-measure is SVM-KL, followed by CFC-JS and SVM-JS, which tied for second place.

Thus we see that JS weighting is suited for centroid classifiers, while KL weighting is more suitable for SVM classifiers. With JS weighting, even CFC based classifiers can beat the performance of a plain unweighted vanilla SVM. This is significant because CFC classifiers are two to three orders of magnitude faster to train compared to SVM classifiers [69].

Figure 4.6: Movie Review Dataset Results

The results for the Sentiment Polarity dataset are shown in Figure 4.7. Here, we see a similar trend where SVM-KL remains the leader in terms of Precision and F-measure, with CFC-JS and CFC-KL in second place, and the group of SVM-IDF, SVM-RF, and SVM-JS roughly tied for third place. Interestingly, CFC-{KL, JS} both had very high recall of 87% and 85% respectively, compared to SVM-KL at 80%. However, all three CFC methods scrape the bottom of the barrel in terms of Precision. If consistent F-measure is desired, SVM-KL is the best performing method for the Sentiment Polarity dataset.

Figure 4.7: Sentiment Polarity Dataset Results

For the Multi-domain Sentiment dataset, the F-measure, recall, and precision results over 4 product domains (book, DVD, electronics, kitchen) are shown in Figure 4.8. From the F-measure results in Figures 4.8, we can see that CFC-KL has the best performance over all domains. CFC-JS is a close second for the book and DVD domains, but retreated to third place for Electronics and Kitchen. In fact, the second best performer is split 50/50 between CFC-JS and SVM-KL, with SVM-JS consistently at fourth place. CFC performed the worst. Note that CFC-KL consistently achieved around 95% F-measure across all domains. SVM-KL is also pretty consistent over all domains, achieving around 91% F-measure. We can conclude from this observation that KL weights not only improve performance, but also help maintain a stable prediction performance across domains.

Figure 4.8: F-Measure for Multi-domain Sentiment Dataset

A View into Centroids

As mentioned, CFC is very aggressive in feature selection; it will assign zero weight to terms appearing in both positive and negative classes, while exponentially boosting the class-exclusive terms. To illustrate, we calculate the percentage of zero term weights in the CFC and CFC-KL centroids. The results are shown in Figure 4.9. We note that CFC-KL has 3% fewer zero-weighted terms compared to CFC on average. The difference between the CFC-KL and CFC in terms of zero term weights is smallest for the kitchen domain, which is the most separable case as mentioned before, and both CFC and CFC-KL prune non-discriminatory terms aggressively. On the other hand, this disparity is very large (5%) for the book domain. This explains why CFC performed the worst for the book domain in terms of recall, as described above.

Figure 4.9: Zero Weighted Terms (%) in each Centroid for the Multi-domain Sentiment Dataset

To understand the growth of the prototype/centroid vectors, we plot the AAC, CFC, and CFC-KL weights of a class-specific centroid term “like” in Figure 4.10, and the same for a generic term “movie” in Figure 4.11. From the two figures, we see that the CFC-KL weight is generally two times larger than AAC, while CFC weights are four times larger than AAC. The CFC weights also fluctuates quite a lot initially compared to the other two weights. The variation in CFC-KL is more stable, while that of AAC is the mildest, maintaining a stable value very early on. Whenever a document containing the centroid word appears, all weights will spike, and then gradually decrease as new documents not containing the word are added.

The interesting thing to note is that all three weights exhibit the same trends, differing only in their raw magnitudes. This means that in theory, we could scale up the weight of either one to obtain the other. The question to ask then is this, which scale gives the best performance? In our results, CFC-KL clearly gives the best results as far as performance and stability are concerned.

Figure 4.10: Evolution of Centroid Term “like”

Figure 4.11: Evolution of Centroid Term “movie”

4.2.4 One-off Classification of Multi-class Imbalanced Data

To study the impact of imbalanced data on the overall performance of classifiers, we evaluate our proposed approach on imbalanced datasets, which are very common in practice. The imbalanced datasets are generated from the 20-newsgroup dataset where we keep one class as positive and the remaining classes as one big negative class. Some details of the modified datasets are listed in Table 4.4, from which we see that each 20-newsgroup sub-dataset has a highly skewed distribution of around 4% positive and 96% negative data.

We apply 10-fold cross validation and test both the CFC approach and our CFC-KL and CFC-JS approaches. The experimental results of the first 10 classes are shown in Figure 4.12 and Figure 4.13 for the 20-newsgroup and Reuters-21578 dataset, respectively.

Table 4.4: Imbalanced 20-Newsgroup and Reuters-21578 Datasets

Dataset (Positive Class)	Terms	N^+	N^-
20-Newsgroup (1)	26,214	799	18,047
(2)	-	973	17873
(3)	-	985	17861
$\vdots$	$\vdots$	$\vdots$	$\vdots$
(8)	-	990	17856
(9)	-	996	17850
(10)	-	994	17852
Reuters-21578 (1)	473,456	3713	4580
(2)	-	2055	6238
(3)	-	321	7972
$\vdots$	$\vdots$	$\vdots$	$\vdots$
(8)	-	114	8179
(9)	-	110	8183
(10)	-	90	8203

For the 20-newsgroup dataset, CFC-JS is the best performer in 7 out of 10 sub-datasets, and a close second for the remaining 3. This shows that CFC-JS is very effective in handling highly imbalanced datasets. CFC is the worst for all except one case, performing up to 30% worse.

Figure 4.12: F-Measure for Imbalanced 20-Newsgroup Dataset

Figure 4.13: F-Measure for Imbalanced Reuters-21578 Dataset

Similar observations can be made for the Reuters-21578 dataset, where CFC-JS leads in 7 out of 10 sub-datasets, with CFC-KL leading the remaining 3. CFC consistently underperform in all but two sub-datasets.

4.2.5 $tf \times JS$ versus $tf \times KL$ on Imbalanced Binary Datasets

Figure 4.14: F1, Precision, Recall for Positive Class on the Book Domain

Figure 4.15: F1, Precision, Recall for Positive Class on the DVD Domain

Figure 4.16: F1, Precision, Recall for Positive Class on the Electronics Domain

Figure 4.17: F1, Precision, Recall for Positive Class on the Kitchen Domain

We study the pros and cons of JS term weighting vis-a-vis KL term weighting. Since we can adjust the class weights $\{\pi_i\}$ in the JS term weighting approach, it can work well on imbalanced dataset. To do that, we generate artificially imbalanced datasets based on the current balanced Multi-domain Sentiment dataset, where the percentage of positive documents varies from 10% to 90%. Figures 4.14 to 4.21 plot the F-measure, precision, and recall of each class ('+':positive, '-' :negative) for all domains of the Multi-domain Sentiment dataset. Clearly, in the highly imbalanced datasets, SVM-JS consistently outperforms SVM-KL. As the dataset approaches equi-balance, SVM-KL is marginally better than SVM-JS. These results are consistent with those reported in Section 4.2.3. Moreover, the performance of SVM-JS is more stable than that of SVM-KL on the imbalanced datasets; SVM-JS performance is quite consistent with respect to the varying amount of positive examples from 10% to 90%.

Figure 4.18: F1, Precision, Recall for Negative Class on the Book Domain

Figure 4.19: F1, Precision, Recall for Negative Class on the DVD Domain

Figure 4.20: F1, Precision, Recall for Negative Class on the Electronics Domain

Figure 4.21: F1, Precision, Recall for Negative Class on the Kitchen Domain

4.3 Summary

In this chapter, we study the theoretical properties of the recently proposed Class Feature Centroid (CFC) classifier by considering the rate of change of each prototype vector with respect to individual dimensions (terms). Borrowing the e-folding time scale function from physics and biology, we show that classical centroid-based classifiers like Arithmetic Average Centroid (AAC) and Cumuli Geometric Centroid (CFC) have an inverse rate of change, while CFC has an exponential rate of change, proportional to the class document frequency. The implication of our theoretical finding is that CFC is inherently biased towards large (dominant majority) classes, which means it is destined to perform poorly for highly imbalanced data. Another practical concern about CFC lies in its overly-aggressive design in weeding out terms that appear in all classes.

To overcome these CFC limitations while retaining its intrinsic and worthy design goals, we proposed an improved and robust centroid-based classifier that uses precise term-class distribution properties instead of simple presence or absence of terms over classes. Specifically, terms are weighted based on the Kullback-Leibler Divergence Measure between pairs of class-conditional term probabilities; we call this the CFC-KL centroid classifier. We then generalized CFC-KL to handle multi-class data by replacing the KL term with the multi-class Jensen-Shannon term, with each class proportion weighted by a soft-max function. We call this generalized approach CFC-JS.

Our proposed approach has been evaluated on 5 datasets. Experimental results show that the CFC-KL method consistently outperforms other methods on binary datasets. For highly imbalanced binary datasets, CFC-JS further outperforms CFC-KL.

To test the individual effectiveness of JS and KL weighting, we further evaluate the performance of SVM classifiers on highly imbalanced binary classification problems. We find that both SVM-JS and SVM-KL achieve significantly better performance for the minority class, making them less susceptible to imbalanced datasets.

In general, we find that CFC when combined with KL (two class) or JS (multi-class) weighting, will significantly boost classification performance, bringing centroid classifiers up to the accuracy level of SVM classifiers. For applications where training time is critical, such as large scale online classification tasks, CFC combined with KL or JS term weighting becomes an attractive option that deserves serious consideration.

Chapter 5

Online Linear Learning

We have proposed supervised term weighting approaches for general text classification in Chapter 3, and centroid classifiers in Chapter 4. The proposed approaches have been shown to work well on text datasets. However, in tasks that require huge number of real-time classifiers, large margin online learning is the only viable option. For example, real-time classification of status/micro-blog updates of millions of social network users requires a personalized online classifier to be maintained for every user [57]. An online learning algorithm updates its decision boundary incrementally after processing each sample. Given a sample, it will first classify it, so called making a prediction. After prediction, the true label is revealed, and the quantitative difference between the prediction and true label is computed as the loss, which is then used to adjust the classifier weights. The goal is to maximize the correctness of future predictions. The proposed approach was evaluated on letter recognition, Web page classification, and sentiment analysis datasets.

As reviewed in Chapter 2, the classical Perceptron [7, 70], Second-order Perceptron (SOP) [15], suite of Passive Aggressive (PA) algorithms [21] and its second order variants Confidence Weighted (CW) learning [26] and Adaptive Regularization Of Weight vectors (AROW) [23], all belong to the same family of online algorithms, which perform well for a variety of real-time applications. However, except for the Second-order Perceptron, these online learning algorithms do not explicitly account for the distribution of the data [15]. PA algorithms that assume the squared Euclidean distance metric work well for data conforming to the Normal distribution. However, for hyper-ellipsoidal data distributions, performance can be marginal.

To overcome this deficiency, we propose using the Mahalanobis distance measure in place of the Euclidean distance in PA. We call the new approach Passive Aggressive Mahalanobis

(PAM). The Mahalanobis distance is slightly more flexible than the Euclidean distance, which assumes a spherical Normal data distribution (diagonal covariance), because it can model ellipsoidal Normal distributions. Given its better adaptability to data, the PAM algorithm should work better than the PA algorithm in practice. PAM's update equation bears a close resemblance to that of CW, which took a different route by assuming that the weights are normally distributed with a mean vector and covariance matrix. However, PAM is different from CW in its update criterion; CW minimizes the differences between the new and old weight distribution whereas PAM simply maintains the original PA goal by minimizing the new and old weight vector differences, constrained by a non-diagonal covariance.

5.1 Passive Aggressive Mahalanobis

The online binary classification framework in this section follows the PA algorithm formulation [21].

5.1.1 Hard Margin PAM

PAM is similar to PA, except for its use of the Mahalanobis distance measure in place of the Euclidean distance measure. The optimization is formulated as

$$\begin{aligned} \mathbf{w}_{t+1} = \operatorname{argmin}_{\mathbf{w} \in \mathcal{R}^n} \frac{1}{2} (\mathbf{w} - \mathbf{w}_t)^T \Sigma_{t-1}^{-1} (\mathbf{w} - \mathbf{w}_t) \\ \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0 \end{aligned} \quad (5.1)$$

where Σ_{t-1} is the covariance matrix of the weight vector distribution at round $t - 1$. Solving the above problem, we have

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \Sigma_{t-1} \mathbf{x}_t \quad \text{and} \quad \tau_t = \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t} \quad (5.2)$$

which we call the hard margin Mahalanobis Passive Aggressive (PAM) as described in Algorithm 5.1.

5.1.2 Soft Margin PAM

Extending PAM to deal with misclassified samples, we introduce the slack variable ξ into the optimization problem

$$\mathbf{w}_{t+1} = \operatorname{argmin}_{\mathbf{w} \in \mathcal{R}^n} \frac{1}{2} (\mathbf{w} - \mathbf{w}_t)^T \Sigma_{t-1}^{-1} (\mathbf{w} - \mathbf{w}_t) + C\xi \quad \text{s.t.} \quad \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \quad \text{and} \quad \xi \geq 0$$

Algorithm 5.1 Passive Aggressive Mahalanobis (PAM).

Input: C = positive aggressiveness parameter**Output:**

None

Process:

```

1: Initialize  $\Sigma_0 \leftarrow \mathbf{I}$ ;  $\mathbf{w}_1 \leftarrow \mathbf{0}$ ;
2: for  $t = 1, 2, \dots$  do
3:   Receive instance  $\mathbf{x}_t \in \mathbb{R}^n$ 
4:   Predict  $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$ 
5:   Receive correct label  $y_t \in \{-1, +1\}$ 
6:   Suffer loss  $\ell_t \leftarrow \max\{0, 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\}$ 
7:   if  $\ell_t > 0$  then
8:     Set  $\tau_t \leftarrow \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t}$  (PAM)
            $\tau_t \leftarrow \min \left\{ C, \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t} \right\}$  (PAM-I)
            $\tau_t \leftarrow \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t + \frac{1}{2C}}$  (PAM-II)
9:   Update  $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \tau_t y_t \Sigma_{t-1} \mathbf{x}_t$ 
            $\Sigma_t \leftarrow \Sigma_{t-1} - \frac{\Sigma_{t-1} \mathbf{x}_t \mathbf{x}_t^T \Sigma_{t-1}}{1 + \mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t}$ 
10:  end if
11: end for

```

where C is the positive aggressiveness constant, which controls the aggressiveness of each update step. The bigger the value of C , the larger the update. We thus have the following closed form solution for the step size.

$$\tau_t = \min \left\{ C, \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t} \right\}$$

Suppose the objective function changes quadratically with the slack variable ξ , we have the following optimization problem.

$$\begin{aligned} \mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathcal{R}^n}{\text{argmin}} \quad & \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T \Sigma_{t-1}^{-1}(\mathbf{w} - \mathbf{w}_t) + C\xi^2 \\ \text{s.t.} \quad & \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \end{aligned} \tag{5.3}$$

Solving the above problem, we have the following result.

$$\tau_t = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t + \frac{1}{2C}}$$

We call these two approaches PAM-I and PAM-II, respectively. Both share the same general form $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \Sigma_{t-1} \mathbf{x}_t$, with a different update step as follows.

$$\tau_t = \min \left\{ C, \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t} \right\} \quad (\text{PAM-I})$$

and

$$\tau_t = \frac{\ell_t}{\mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t + \frac{1}{2C}} \quad (\text{PAM-II})$$

5.1.3 Covariance Matrix Estimation

PAM-I and PAM-II involve computation of a covariance matrix Σ over the weight vector $\mathbf{w}$, which can be approximated as follows. Consider the evolution of the objective function starting at $t = 0$ and $\Sigma_0 = \mathbf{I}$. At round T , we have the loss function $1 - y_T \mathbf{w}_T \cdot \mathbf{x}_T$ where $\mathbf{w}_T$ is the weight vector at round T . Denoting $\mathbf{y}$ as a vector $[y_1, y_2, \dots, y_T]$ and $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_T]$ as a matrix of column input vectors, we can write the PAM weight update as

$$1 - \mathbf{y} \mathbf{X} \mathbf{w}_T = 0 \quad \text{or} \quad \mathbf{y} - \mathbf{X} \mathbf{w}_T = \mathbf{0} \quad (5.4)$$

Multiplying the above equality by $\mathbf{X}^T$, we have

$$\mathbf{X}^T (\mathbf{y} - \mathbf{X} \mathbf{w}_T) = \mathbf{0} \quad \text{or} \quad \mathbf{X}^T \mathbf{X} \mathbf{w}_T = \mathbf{X}^T \mathbf{y} \quad (5.5)$$

Multiplying the above equality by the pseudo-inverse $(\mathbf{X}^T \mathbf{X})^{-1}$, we have

$$\mathbf{w}_T = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (5.6)$$

Assuming an independent and identically distributed noise model, we can write $\mathbf{y} = \mathbf{X} \mathbf{w} + \mathbf{e}$, where $\mathbf{e}$ is an error vector. Substituting $\mathbf{y}$ into the above equality, we have

$$\mathbf{w}_T = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{X} \mathbf{w} + \mathbf{e}) = \mathbf{w} + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{e} \quad (5.7)$$

The covariance matrix of $\mathbf{w}_T$ has the form

$$\Sigma_T = \text{Cov}(\mathbf{w}_T) = \mathbb{E}[(\mathbf{w}_T - \mathbf{w})(\mathbf{w}_T - \mathbf{w})^T] \quad (5.8)$$

or

$$\begin{aligned} \Sigma_T &= \mathbb{E}[(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{e} \mathbf{e}^T \mathbf{X} (\mathbf{X} \mathbf{X}^T)^{-1}] \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbb{E}[\mathbf{e} \mathbf{e}^T] \mathbf{X} (\mathbf{X} \mathbf{X}^T)^{-1} \end{aligned} \quad (5.9)$$

Since the PAM weights are updated to achieve a margin of at least 1, we can assume that $E[\mathbf{e}\mathbf{e}^T] = \sigma^2 \mathbf{I}$. The covariance matrix can thus be approximated as follows.

$$\Sigma_T = \sigma^2 (\mathbf{X}\mathbf{X}^T)^{-1} \simeq (\mathbf{X}\mathbf{X}^T)^{-1} \quad (5.10)$$

Following the work of [15], we can write

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + \mathbf{x}_t \mathbf{x}_t^T. \quad (5.11)$$

Applying the Sherman-Morrison formula to (5.11), we have

$$\Sigma_t = \Sigma_{t-1} - \frac{\Sigma_{t-1} \mathbf{x}_t \mathbf{x}_t^T \Sigma_{t-1}}{1 + \mathbf{x}_t^T \Sigma_{t-1} \mathbf{x}_t} \quad (5.12)$$

From Equation (5.11), we can conclude that $\Sigma_t^{-1} \succeq \Sigma_{t-1}^{-1}$ and $\Sigma_t \preceq \Sigma_{t-1}$. Like the PA family, all three PAM algorithms share the same weight update equation, differing only in the update rate τ_t , as shown in Algorithm 5.1. In fact, the PAM-II weight update resembles that of Adaptive Weight Regularization [23] (AROW). The difference between PAM-II and AROW is that PAM-II does not explicitly regulate the update of the covariance matrix Σ_t . This is because PAM was primarily motivated by adding a data noise model to PA, while AROW and CW started out by assuming a distribution of weights. The end results are very similar, differing only in the update rates.

Ma et al. [61] examined in depth several strategies to estimate the covariance matrix efficiently, along with their practical implications specifically for CW, which can be used for any second-order learning algorithms, including PAM. In this chapter, we will not focus on the practical issue of computing the covariance, but instead measure the classification performance of both CW and PAM assuming that a full covariance matrix is obtainable for moderate number of input dimensions.

5.1.4 PAM Error Analysis

In this section we provide several theoretical results for PAM.

Theorem 5.1 (Relative Loss Bound). Given a sequence of M examples $[(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_M, y_M)]$, any weight vector $\mathbf{u} \in \mathbb{R}^n$, and loss $\ell_t^* = 0$ for all t , the cumulative relative loss of PAM is upper bounded by

$$\sum_t \ell_t^2 \leq (\|\mathbf{u}\|^2 + \mathbf{u}^T \mathbf{X}_U \mathbf{X}_U^T \mathbf{u}) \max_t \mathbf{x}_t^T \Sigma_t \mathbf{x}_t \quad (5.13)$$

where U is the set of indices for examples leading to the weight updates and $\mathbf{X}_U \mathbf{X}_U^T = \sum_{t \in U} \mathbf{x}_t \mathbf{x}_t^T$.

Theorem 5.2 (PAM-I Mistake Bound). Given a sequence of examples $[(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_M, y_M)]$ and any weight vector $\mathbf{u} \in \mathbb{R}^n$, the number of mistakes made by PAM-I is upper bounded by

$$\max\{\max_t \mathbf{x}_t^T \Sigma_t \mathbf{x}_t, \frac{1}{C}\} \left(\|\mathbf{u}\|^2 + \mathbf{u}^T \mathbf{X}_U \mathbf{X}_U^T \mathbf{u} + C \sum_{t=1}^M \ell_t^* \right) \quad (5.14)$$

where C is a positive aggressiveness parameter.

Theorem 5.3 (PAM-II Loss Bound). Given a sequence of examples $[(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_M, y_M)]$ and any weight vector $\mathbf{u} \in \mathbb{R}^n$, the cumulative relative loss of PAM-II is upper bounded by

$$\sum_t \ell_t^2 \leq \left(\max_t \mathbf{x}_t^T \Sigma_t \mathbf{x}_t + \frac{1}{2C} \right) \left(\|\mathbf{u}\|^2 + \left(\frac{2}{1 + \frac{1}{2C}} \right) \mathbf{u}^T \mathbf{X}_U \mathbf{X}_U^T \mathbf{u} + C \sum_{t=1}^M (\ell_t^*)^2 \right) \quad (5.15)$$

In [21], the squared upper loss bound of the PA algorithm was defined as $\|\mathbf{u}\|^2 (\max |\mathbf{x}|)^2$. This bound depends only on the norm of weight vector $\mathbf{u}$. It does not consider the input data distribution while the upper bound of the PAM algorithm depends on both the norm of $\mathbf{u}$ and $\mathbf{u}^T \mathbf{X}_U \mathbf{X}_U^T \mathbf{u}$, the data spectral term. We know that the second term is finite and bounded by the maximal eigenvalues of the matrix $\mathbf{X}_U \mathbf{X}_U^T$. Another term in the loss bound of the PAM-I and the PAM-II algorithms is $\mathbf{x}_t^T \Sigma_t \mathbf{x}_t$, which is a trade-off factor between the hinge-loss term and the data spectral term. In CW learning, the matrix Σ_t is called the confidence, which decreases monotonically with the observed data. Given that $\Sigma_0 = I$, the inequality $\mathbf{x}_t^T \Sigma_t \mathbf{x}_t \leq \mathbf{x}_t^T \mathbf{x}_t$ will always hold, which causes the data spectral term to increase with the hinge-loss quantity. However, it is very difficult to compare the upper loss bounds of the two families because both depend on the input distribution. In other words, we cannot conclude that one is tighter than the other.

5.2 Performance Evaluation

5.2.1 Datasets

A total of 6 datasets were used including two binary classification datasets (CRX and BUPA datasets from UCI [3]), two binary web datasets (WebKB and Twitter Sentiment), and two multi-class datasets (USPS and MNIST). For the multi-class datasets, one random class out of C classes was selected as positive, and a negative class of equal size was generated by

sampling (the same number of samples as the positive class) from the remaining $C - 1$ classes. Where applicable, all experiments were repeated 10 times with different randomizations, and the average results shown/plotted. The results on the twitter dataset were single-run, since the data are deterministically ordered and binary.

Table 5.1: Datasets.

Name	Instances	Features	Classes
BUPA	345	6	2
CRX	653	47	2
USPS	7291	256	10
MNIST	60000	783	10
WebKB	1051	3000	2
Twitter	3107	1520	2

5.2.2 Cumulative Error Rate

We used the standard cumulative error rate, which is the ratio of mistakes over the total number of examples. To ensure a fair comparison of our proposed algorithms with the original Passive Aggressive algorithms, we grid-searched for the optimal aggressiveness parameter C in all PA-based algorithms. We excluded the PA results as it performed worse than PA-I and PA-II [21]. Also, PA-I is slightly worse than PA-II [21] therefore we only showed the cumulative error rate comparisons between PA-II, PAM-II, and CW here.

Figure 5.1: Cumulative Error Rate for BUPA and CRX.

The cumulative error rates on the two binary datasets are shown in Figure 5.1. For BUPA, all three algorithms started off with similar loss, but PAM-II starts to pull away from the pack after 60 examples, and consistently exhibits lower log-mistake rates thereafter. For CRX, PAM-II leads after iteration 30, with an overall lower mistake rate thereafter.

Figure 5.2: Cumulative Error Rate for USPS and MNIST.

Figure 5.2 shows the cumulative error rates on the two letter recognition datasets. PAM-II performed better right from the start, but not significantly better overall because the negative class in this case is heterogenous (formed by a uniform equal-sized sample of the non-positive class); better results can be expected in a one-of classification.

The WebKB dataset contains 1051 web documents from two classes, each with two views. We tested all algorithms only on the textual view, with results shown in Figure 5.3. Again, PAM achieved consistently lower log-mistake rates, widening the gap with increasing number of examples.

Figure 5.3: Cumulative Error Rate for WebKB.

5.2.3 Classification Accuracy

In practice, classification performance in terms of F-measure is typically more important than cumulative error rates. The positive (+) and negative (-) class F-measures for all 5 datasets are listed in Table 5.2 with the best results in bold. PAM-II consistently outperformed the other algorithms, including CW. Although, CW performed better than the original PA-I and PA-II, it still falls a little behind PAM. For BUPA, PAM-II is more than 2% better than CW. PAM-II achieved the largest winning margin against CRX+, where it is more than 11% better than PA-II, and 2% better than CW. Overall, PAM-II beats CW only by a 1-2% margin. Again, the positive class improvements over PA-II are significantly better because it is much more homogeneous compared to the artificially consolidated negative class.

Table 5.2: F1 (%) for Positive (+) and Negative (-) Classes.

Dataset	PA-II	PAM-II	CW
BUPA+	61.21 $\pm$ 2.78	64.28 $\pm$ 1.13	61.79 $\pm$ 2.43
BUPA-	57.85 $\pm$ 1.31	60.20 $\pm$ 2.55	57.61 $\pm$ 2.00
CRX+	68.54 $\pm$ 0.40	80.37 $\pm$ 0.51	78.28 $\pm$ 0.93
CRX-	77.83 $\pm$ 0.62	84.13 $\pm$ 0.62	82.54 $\pm$ 0.60
USPS+	94.08 $\pm$ 2.77	95.26 $\pm$ 2.56	94.81 $\pm$ 2.79
USPS-	93.84 $\pm$ 2.78	95.05 $\pm$ 2.56	94.57 $\pm$ 2.79
MNIST+	57.85 $\pm$ 3.43	59.51 $\pm$ 1.72	58.64 $\pm$ 0.90
MNIST-	58.97 $\pm$ 2.47	60.47 $\pm$ 1.48	58.52 $\pm$ 1.12
WEBKB+	77.02 $\pm$ 1.14	78.90 $\pm$ 1.63	76.49 $\pm$ 1.40
WEBKB-	90.98 $\pm$ 0.66	92.94 $\pm$ 0.84	90.78 $\pm$ 0.74

5.2.4 Online Microblog Data

To illustrate the utility of online algorithms, we apply them to learn emotions from real-life micro-blogs. The *Twitter*¹ sentiment dataset [57] is a collection of micro-blogs (tweets) written by 6 users. Each tweet is manually labeled as emotional (positive) or non-emotional (negative). An online model was applied to each user's tweets in chronological sequence. Each model was initialized to some random weights; after it classifies an incoming tweet, the tweet's true label is revealed to update the model weights, and the online classification/learning continues until the last tweet is predicted.

¹<http://twitter.com>

Figure 5.4: Cumulative Error Rate for the Twitter Sentiment Dataset.

From the individual loss plots in Figure 5.4, PAM-II again consistently outperformed the other algorithms. However, the advantage of PAM-II depends very much on the dataset. For instance, for user DenyceLawton in Figure 5.4(c), PAM-II did significantly better than the others but for another user CarlaMedina in Figure 5.4(b), PAM-II performed only marginally better. On closer examination, we found that CarlaMedina writes equally frequently in Spanish and English. Since our human labeler is not Spanish literate, a large portion of the tweets

have been labeled incorrectly. For example, Spanish emotions were not properly labeled, labeled emotional tweets contain a mix of Spanish and English with English terms acting as the decisive factor. As a result, the labeling for user CarlaMedina is very noisy. Another consequence of not knowing the language is the highly imbalanced class distribution, with user CarlaMedina having the smallest raw count of 250 positive (emotional) labeled samples. Specifically, users AudreyWalker, CarlaMedina, DenyceLawton, IheartBrooke, RealMichelleW, and SabrinaBryan have 15.5%, 19.4%, 41.6%, 18.9%, 17.0%, and 17.4% positive tweets respectively. For PAM-II, this means that the model would have very little chance to make a wrong prediction and significantly adjusting its weight; an occasional positive sample would cause the margin to be reduced. For such a case, PAM does not benefit much from considering the sample distribution, since the covariance matrix would account for a far smaller number of samples

For CarlaMedina, PAM II started to decisively outperform CW only after around 250 samples, by when it should have seen approximately 50 (19.4% of 250) positive samples, assuming a uniform class distribution. For other marginal users like AudreyWalker (483 positive, PAM wins after 40 samples), RealmichelleW (495 positive, PAM wins after 100), and SabrinaBryan (551 positive, PAM wins after 200), who all have around 500 total raw positive tweets, their cumulative PAM loss rates were all able to pull away from the competitor earlier than CarlaMedina (342 positive, PAM wins after 250), simply because they have a larger number of positive tweets.

5.3 Summary

In this chapter, we proposed PAM, a generalization of the Passive Aggressive algorithms [21] that takes into account the data spectral properties. PAM was evaluated on several datasets and found to consistently outperform other online algorithms, including its cousin Confidence Weighted learning. Results on online classification tasks have shown an average of 4% to 12% improvements in F1-measure. We have also validated the practicality and superiority of PAM on a real-world twitter emotion classification dataset.

Compared to PA, PAM runs slower because it needs to compute the covariance matrix, which scales quadratically with the number of features. To solve this problem, we can deploy the approximate version of the PAM algorithm by calculating the diagonal matrix in the same way as the CW algorithm [26, 61].

Chapter 6

Two-View Online Linear Learning

In the previous chapter, we proposed an online learning algorithm (PAM), which was shown to work better than existing online algorithms on large datasets. However, PAM does not scale well to high-dimension datasets due to its exponential computational requirement of the covariance matrix. Further, real world datasets may contain multiple views of disparate dimensions, where the larger views are likely to dominate the smaller ones. Therefore, in this chapter, we propose a two-view learning algorithm that utilizes two different views of the same data to achieve something that is better than the sum of its parts. For both tractability and scalability reasons, we drop the covariance matrix of PAM and fall back to good old PA when formulating the two-view variation. We then evaluated our two-view online PA learning on document classification datasets such as Web page, products review, Ads, and math document datasets.

In two-view online learning, training data are triplets $(\mathbf{x}_t^A, \mathbf{x}_t^B, y_t) \in \mathbb{R}^n \times \mathbb{R}^m \times [-1, +1]$, which arrive in sequence where $\mathbf{x}_t^A \in \mathbb{R}^n$ is the first view vector, $\mathbf{x}_t^B \in \mathbb{R}^m$ is the second view vector, and y_t is their common label. The goal is to learn the coupled weights $(\mathbf{w}_t^A, \mathbf{w}_t^B)$ of a *hybrid model* [57] defined as follows:

$$f(\mathbf{x}_t^A, \mathbf{x}_t^B) = \text{sign}\left(\eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1 - \eta) \mathbf{w}_t^B \cdot \mathbf{x}_t^B\right)$$

where $\eta \in (0, 1)$ is used to adjust the relative importance of each view.

Let $g(\mathbf{x}_t^A, \mathbf{x}_t^B) = \eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1 - \eta) \mathbf{w}_t^B \cdot \mathbf{x}_t^B$. To incorporate the new model into the algorithm, we define the loss function as follows:

$$\ell((\mathbf{w}_t^A, \mathbf{w}_t^B); (\mathbf{x}_t^A, \mathbf{x}_t^B, y_t)) = \begin{cases} 0 & \text{if } g(\mathbf{x}_t^A, \mathbf{x}_t^B) \geq 1 \\ 1 - g(\mathbf{x}_t^A, \mathbf{x}_t^B) & \text{otherwise} \end{cases} \quad (6.1)$$

6.1 Math Feature Extraction

Due to the existence of both semantic and structural information, the preprocessing step for math expressions is more complex than that of text documents. For the purpose of math topic classification, the extracted math features should be representative enough to reflect the underlying characteristics of each math topic. As such, we perform the following two steps to extract math features.

- Content MathML conversion. We first convert math expressions from $\text{\LaTeX}$ into the Content MathML format.
- Math feature extraction. We then extract math features by traversing the MathML tree.

Listing 6.1: *MathML* Content Markup of $(x + y)^2$

```
<apply>                % apply operator

  <power/>              % power operator
  <mfence>
    <apply>              % apply operator
      <plus/>            % plus operator
      <ci>x</ci>         % variable (ci) x
      <ci>y</ci>         % variable (ci) y
    </apply>
  </mfence>
  <cn>2</cn>            % constant (cn) 2
</apply>
```

To convert math expressions from $\text{\LaTeX}$, we use the SnuggleTeX library¹. We first convert the math expressions from $\text{\LaTeX}$ to the representation MathML format, then we use cascading stylesheets to map the representation MathML to content MathML. MathML is selected over $\text{\LaTeX}$ for its rich semantics and ease of processing via standard XML libraries. Listing 6.1 shows an example of content MathML for the math expression $(x + y)^2$.

For content MathML data, we use the XML tree traversal approach for extracting math features. Here we only use two kinds of features, single features and combination features. The single features are used to express constant numbers, variable names, functions names, etc. Combination features are the combinations of math operators and operands in the math expressions.

¹<http://www2.ph.ed.ac.uk/snuggletex>

Take the sub-expression $x + y$ in Listing 6.1 as an example, based on the content MathML data, we have two single features cix and ciy , where ci stands for a variable and cix stands for a variable named x . We also have one combination feature $pluscixciy$, where $plus$ stands for the operator $+$ and $pluscixciy$ denotes the operator $+$ applied to two operands x and y .

6.2 Two-view Passive Aggressive Algorithm

6.2.1 Relationship between Views

To determine the relatedness between two views, we define a disagreement factor as follows:

$$|\eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A - (1 - \eta) \mathbf{w}_t^B \cdot \mathbf{x}_t^B| \quad (6.2)$$

where $|\cdot|$ denotes the absolute function and η , similar to the hybrid model, is used to trade off the disagreement between the two views. The objective is to minimize the disagreement between the two views.

Based on the above relatedness measurement, we define a hybrid prediction function to predict the label as follows:

$$f(\mathbf{x}_t^A, \mathbf{x}_t^B) = \text{sign}(\eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1 - \eta) \mathbf{w}_t^B \cdot \mathbf{x}_t^B)$$

In this case, we do not really care whether $f(\mathbf{x}_t^A)$ or $f(\mathbf{x}_t^B)$ can individually classify the example correctly. What we want instead is for their weighted sum $f(\mathbf{x}_t^A, \mathbf{x}_t^B)$ to correctly predict the class label.

The ideal objective function should include both the new loss function in Equation (6.1) and the view relatedness factor in Equation (6.2). Similar to the PA algorithm, the new weights of the two-view learning algorithm are updated based on the following optimization problem,

$$\begin{aligned} (\mathbf{w}_{t+1}^A, \mathbf{w}_{t+1}^B) &= \underset{(\mathbf{w}^A, \mathbf{w}^B) \in \mathbb{R}^n \times \mathbb{R}^m}{\text{argmin}} \quad \frac{1}{2} \|\mathbf{w}^A - \mathbf{w}_t^A\|^2 + \frac{1}{2} \|\mathbf{w}^B - \mathbf{w}_t^B\|^2 \\ &\quad + \gamma |\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B| + C\xi \\ \text{s.t.} \quad &1 - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B \leq \xi \\ &\xi \geq 0 \end{aligned}$$

where γ and C are positive agreement and aggressiveness parameters, respectively. While γ is used to adjust the importance of the agreement between the two views, C is used to control the aggressiveness property of the PA algorithm. Note that the multiplier y_t in the agreement is there just for the convenience of subsequent derivation.

The absolute function can be written for all x, y as $|x - y| = \max(x - y, y - x)$. Therefore, we have

$$|\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B| = \max \left(\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B, (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A \right)$$

Suppose $z = |\eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B|$, the above optimization problem can be rewritten as follows.

$$\begin{aligned} (\mathbf{w}_{t+1}^A, \mathbf{w}_{t+1}^B) = \underset{(\mathbf{w}^A, \mathbf{w}^B) \in \mathbb{R}^n \times \mathbb{R}^m}{\operatorname{argmin}} \quad & \frac{1}{2} \|\mathbf{w}^A - \mathbf{w}_t^A\|^2 + \frac{1}{2} \|\mathbf{w}^B - \mathbf{w}_t^B\|^2 + \gamma z + C\xi \\ \text{s.t.} \quad & 1 - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B \leq \xi; \\ & \xi \geq 0; \\ & z \geq \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B; \\ & z \geq (1 - \eta) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B - \eta y_t \mathbf{w}^A \cdot \mathbf{x}_t^A. \end{aligned} \tag{6.3}$$

Remark. It is clear that the above optimization problem can effectively address the relatedness of the two views. When the value of the objective function is small, the disagreement factor z of the two views is forced to be small. If we set $\gamma = 0$, the problem reduces to learning two linear models independently; if we set $\gamma = 1$, we aggressively penalize the disagreement of the two views. By adjusting the value of $0 < \gamma < 1$, we can calibrate the collaboration between the two views.

Proposition 6.1. The optimization problem in Equation (6.3) has the following closed-form solution,

$$\mathbf{w}^A = \mathbf{w}_t^A - \eta(\alpha - \beta - \tau) y_t \mathbf{x}_t^A$$

and

$$\mathbf{w}^B = \mathbf{w}_t^B - (1 - \eta)(\beta - \alpha - \tau) y_t \mathbf{x}_t^B$$

where

$$\begin{aligned} \tau &= \min \left\{ C, \frac{(\alpha - \beta) \left(\eta^2 \|\mathbf{x}_t^A\|^2 - (1 - \eta)^2 \|\mathbf{x}_t^B\|^2 \right) + \ell_t}{\eta^2 \|\mathbf{x}_t^A\|^2 + (1 - \eta)^2 \|\mathbf{x}_t^B\|^2} \right\}, \\ \alpha &= \min \left\{ \gamma, \frac{1}{2} \left(\gamma + \frac{1}{\eta} \frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\|\mathbf{x}_t^A\|^2} - \frac{1}{1 - \eta} \frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\|\mathbf{x}_t^B\|^2} \right) \right\}, \text{ and} \\ \beta &= \min \left\{ \gamma, \frac{1}{2} \left(\gamma - \frac{1}{\eta} \frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\|\mathbf{x}_t^A\|^2} + \frac{1}{1 - \eta} \frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\|\mathbf{x}_t^B\|^2} \right) \right\}. \end{aligned}$$

In these formulas, τ , α , and β are intermediate variables, whose combinations $\alpha - \beta - \tau$ and $\beta - \alpha - \tau$ denote the learning rates of the two views, respectively. Finally, we obtain our Two-view Passive Aggressive formulation as shown in Algorithm 6.2.

Algorithm 6.2 Two-view Passive Aggressive Algorithm**Input:**

C = positive aggressiveness parameter
 γ = positive agreement parameter
 η = positive view weight parameter

Output:

None

Process:

Initialize $\mathbf{w}_1^A \leftarrow \mathbf{0}$; $\mathbf{w}_1^B \leftarrow \mathbf{0}$;
for $t = 1, 2, \dots$ **do**
 Receive instances $\mathbf{x}_t^A \in \mathbb{R}^n$ and $\mathbf{x}_t^B \in \mathbb{R}^m$
 Predict $\hat{y}_t = \text{sign}(\eta \mathbf{w}_t^A \cdot \mathbf{x}_t^A + (1 - \eta) \mathbf{w}_t^B \cdot \mathbf{x}_t^B)$
 Receive correct label $y_t \in \{-1, +1\}$
 Suffer loss
 $\ell_t \leftarrow \max \left\{ 0, 1 - \eta y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A - (1 - \eta) y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B \right\}$
 if $\ell_t > 0$ **then**
 Set
 $\alpha \leftarrow \max \left\{ 0, \min \left\{ \gamma, \frac{1}{2} \left(\gamma + \frac{1}{\eta} \frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\|\mathbf{x}_t^A\|^2} - \frac{1}{1 - \eta} \frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\|\mathbf{x}_t^B\|^2} \right) \right\} \right\}$
 $\beta \leftarrow \max \left\{ 0, \min \left\{ \gamma, \frac{1}{2} \left(\gamma - \frac{1}{\eta} \frac{y_t \mathbf{w}_t^A \cdot \mathbf{x}_t^A}{\|\mathbf{x}_t^A\|^2} + \frac{1}{1 - \eta} \frac{y_t \mathbf{w}_t^B \cdot \mathbf{x}_t^B}{\|\mathbf{x}_t^B\|^2} \right) \right\} \right\}$
 $\tau_t \leftarrow \min \left\{ C, \frac{(\alpha - \beta)(\eta^2 \|\mathbf{x}_t^A\|^2 - (1 - \eta)^2 \|\mathbf{x}_t^B\|^2) + \ell_t}{\eta^2 \|\mathbf{x}_t^A\|^2 + (1 - \eta)^2 \|\mathbf{x}_t^B\|^2} \right\}$
 Update
 $\mathbf{w}_{t+1}^A \leftarrow \mathbf{w}_t^A - \eta(\alpha - \beta - \tau_t) y_t \mathbf{x}_t^A$
 $\mathbf{w}_{t+1}^B \leftarrow \mathbf{w}_t^B - (1 - \eta)(\beta - \alpha - \tau_t) y_t \mathbf{x}_t^B$
 end if
end for

6.2.2 Getting Rid of Parameter η

One limitation of the Two-view PA algorithm (Algorithm 6.2) is that its view parameter η must be chosen beforehand. In practice, however, choosing a suitable value for this parameter can be time-consuming. In addition, the optimal value may change with time, thereby affecting the performance. Therefore, we propose an adaptive variant of the Two-view PA algorithm that automatically determines the best value of η . The idea is to modify the objective function of the optimization problem in Equation (6.3) by adding a new regularization factor $\frac{\zeta}{2}(\eta - \eta_t)^2$.

Proposition 6.2. The new optimization problem has the closed-form solution as follows:

$$\eta = \eta_t - \frac{1}{\zeta} \left((\alpha - \beta - \tau) y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - (\beta - \alpha - \tau) y_t \mathbf{w}^B \cdot \mathbf{x}_t^B \right) \quad (6.4)$$

Initially the two views are treated equally, i.e., $\eta = 0.5$, and will be updated based on

Equation (6.4) thereafter. This is thus an adaptive version of the Two-view PA algorithm, which we call the Adaptive Two-view PA algorithm.

6.3 Performance Evaluation

6.3.1 Datasets

We evaluate the online classification performance of our proposed Adaptive Two-view PA algorithm on three benchmark datasets, Ads [50], Product Review [57], and WebKB [81], and a math dataset taken from the Math Overflow site. The *single-view* PA algorithm serves as the baseline. We employ a different PA model for each view, denoted as *PA View 1* and *PA View 2* for each view, respectively. We also report the results from a simple concatenation of the input feature vectors from each view to form a larger feature set, denoted as *PA Cat*. We compare the performance of all algorithms based on F-measure instead of accuracy because most of the datasets are highly (class) imbalanced.

Table 6.1: Summary of Datasets in Our Experiments

	View		Sample Count		
	Name	#Dim	#Pos	#Neg	#Total
Ads	img & dest	929	459	2820	3279
	alt & base	602			
Product Review	lexical	2759	1000	1000	2000
	formal	5			
WebKB	page	3000	230	821	1051
	link	1840			

6.3.2 Two-view Learning Evaluation

In this section, we evaluate the proposed algorithm on three benchmark datasets, whose characteristics are listed in Table 6.1. We use cross validation to select the optimal value for all parameters C , γ , and η , so as to make the comparison fair and meaningful. In our Two-View PA approach, η is learnt using the Adaptive Two-view PA algorithm.

Table 6.2: F-Measure on 3 Benchmark Datasets

Dataset	PA View 1	PA View 2	PA Cat	Two-view PA
Ads	83.41 ± 2.76	76.26 ± 2.21	81.95 ± 2.55	84.96 ± 2.41
Product Review	86.69 ± 1.39	70.80 ± 1.57	86.68 ± 1.63	88.72 ± 1.80
WebKB	93.76 ± 2.30	90.68 ± 2.70	95.56 ± 0.98	98.02 ± 2.14

The Ads dataset was first used by Kushmerick [50] to automatically filter advertisement

images from web pages. In our experiments, we used just four views, namely *image URL view*, *destination URL view*, *base URL view*, and *alt view*. Since we are limited to handling two views for each task, the first and second views were concatenated into View 1 and the remaining two views were concatenated into View 2. This dataset has 3279 examples, including 459 positive examples (ads), with the remaining samples negative (non-ads). Experimental learning results on the Ads dataset are shown in Table 6.2, which shows the proposed algorithm to have the best F-measure score, performing more than 1% better than the runner-up, PA View 1. Interestingly, PA Cat fared worse than PA View 1, which could be due to a noisy decision boundary in the space of PA View 2. This can also be seen by the marginal improvement of 1% of the Two-view PA results.

The Product Review dataset is crawled from popular online Chinese cell-phone forums [57]. The dataset has 1000 true reviews and 1000 spam reviews. It comprises two sets of features: one based on review content (*lexical view*) and the other based on extracted characteristics of the review sentences (*formal view*). The experimental results on this dataset are shown in Table 6.2. Again, our Two-view PA performs better than the rest, beating the runner-up (PA View 1) by 2%. Here PA Cat performed better than either view alone, which is typically the case.

The WebKB course dataset has been frequently used in the empirical study of multi-view learning. It comprises 1051 web pages collected from the computer science departments of four universities. Each page has a class label, course or non-course. The two views of each page are the textual content of a web page (*page view*) and the words that occur in the hyperlinks of other web pages pointing to it (*link view*), respectively. We used a processed version of the WebKB course dataset [81] in our experiment. The performance of PA Cat here is also better than the best single-view PA. And the Two-view PA performed more than 2% better than PA Cat, and 4% better than the best individual view PA.

6.3.3 Math Topic Classification

We downloaded more than 30,000 math questions and answers belonging to 20 math categories such as algebraic geometry, number theory, algebraic topology, combinatorics, group theory, probability, etc., from the Math Overflow site, a popular math question answering system. Each math question or answer is treated as one math document, which may contain both text content and math expressions. Since the limitation of the two-view classification algorithms is that they only work on binary datasets, to evaluate math topic classification,

we choose two major categories (algebraic geometry and number theory), which comprises more than 7,200 math documents. Some sample documents are shown in Table 6.3. After preprocessing, we obtain the following datasets:

Table 6.3: Math Overflow Sample Documents

Algebraic Geometry	Number Theory
<p>I've realized that my answer is wrong. Here's the right answer: if Z is a normal scheme and $f : Z \rightarrow X$ is a morphism such that each associated point of Z maps to an associated point of X, then f factors through n.</p> <p>A counterexample that shows why what I said previously does not work: let f be the inclusion of the node into the nodal curve. There is no unique lift of f to the normalization of the nodal curve...</p>	<p>Given a topological ring R, under what conditions and in what way, can one induce a topology on the R-points of a scheme X? For example, if X is P^n or A^n, one has natural topology on the R-points.</p> <p>If G is a group scheme/A and R is A-algebra (still a topological ring), will the induced topology on $G(R)$ (as above) automatically make G into a topological group...</p>
<p>Normalization is right adjoint to the inclusion functor from the category of normal schemes into the category of reduced schemes. In other words, if $n : Y \rightarrow X$ is the normalization of X and $f : Z \rightarrow X$ is any morphism where Z is a normal scheme, then f factors uniquely through n.</p>	<p>Brian Conrad has some notes on this on his website ("Some notes on topologizing the adelic points of schemes, unifying the viewpoints of Grothendieck and Weil"). The short version is that if X is affine, you can topologize $X(R)$ in a natural, functorial way...</p>
<p>I think it's okay if you rephrase like this: let X be a scheme with ring of fractions R. Then the normalization of X is the universal example of a normal scheme with ring of fractions R mapping to X...</p>	<p>For adelic points of X (or G), one can first topologize $X(Q_p)$ so that it becomes a p-adic analytic variety, and for almost all p one can define an open subset $X(Z_p)$. Then take $X(A)$ to be the restricted product...</p>
...	...

- Text only. All math expressions are removed from math documents. The remaining text-only documents are transformed into a vector format using $tf \times idf$ weighting [62].
- Math only. To evaluate whether math expressions are useful for math topic classification, we extract all math expressions from each math document. Then we apply the math feature extraction method of Section 6.1 to generate the *math only* dataset.
- Raw. It is trivial to treat math expressions as normal text data. One can use the latest text preprocessing techniques to extract textual features from both math and text.

- Math and text. We store the *math only* and *text only* datasets as two datasets (views), called the *math & text* dataset.
- Key phrase. Math key phrases extracted using a NLP (Natural Language Processing) tool to perform noun phrase extraction.

PA Only

After preprocessing, we then run the PA algorithm on all datasets using 5-fold cross validation. The experimental results are shown in column 2 of Table 6.4. Note that the PA model trained on Math & Text operates on a concatenation of the two views. We see that the PA algorithm performs the worst on the *math only* dataset and best on the *text only* dataset. Clearly, there is much room for improvement in our math expression extraction process.

For the *raw text* and *math & text* datasets, the F-measure of the PA algorithm is not high, although both math and text data are taken into consideration. In fact, the PA algorithm did very poorly on just the *raw text*. Its performance is only better than its *math only* dataset results. Compared with the *math only* and *raw text* datasets, the *math & text* dataset can improve the performance of the PA algorithm. However, its performance on this dataset is actually worse than the *text only* dataset.

The Missing View

In practice, math documents do not always contain both text data and math expressions. So what happens if either text data or math expressions are available, but not both? Can Two-view PA work in this case? To find out, we trained the Two-view PA on the *text & math* dataset and tested it on the *text only* and *math only* datasets. It means that we trained the Two-view PA on both views to have two weight vectors and then used them to predict the labels for documents in individual views. While testing on one view, we will ignore the other view.

We also ran the Two-view PA algorithm on the *math & text* dataset (treating each as one view), whose 75.70% F-measure score is more than 7% better than the PA algorithm (68.91%), which was trained on the combined view. Moreover, compared with the PA on *text only* dataset (72.73%), the two view performance (75.70%) is improved by nearly 3%.

This means that when user posts a math question containing math expressions but without text data, the Two-view PA algorithm performs better than the PA algorithm trained on the *math only* dataset by up to 7%. The results are encouraging because we are able to take

advantage of the data of one view to improve the performance of the classifier on another view by enforcing agreement between the two views.

Table 6.4: F-Measure Comparison on Math Overflow Datasets (* trained on the Math & Text Dataset, with results for individual views shown)

Dataset	PA	Adaptive Two-view PA*
Text Only	72.73 ± 2.97	73.85 ± 2.90
Math Only	56.31 ± 7.47	64.25 ± 6.05
Raw	61.02 ± 0.12	-
Math & Text	68.91 ± 5.03	75.70 ± 3.37
Key Phrase	76.78 ± 0.90	78.15 ± 1.27

6.3.4 View Weight Parameter Learning

We investigate ways to find the optimal value of η for the Adaptive Two-view PA algorithm for all datasets. The experimental results are listed in Table 6.5. Note that for all 3 datasets, View 1 performed better than View 2, individually. For the Ads and WebKB datasets, we note that $\eta < 0.5$ despite View 1 performing better than View 2. On the other hand, for the Math Overflow and Product Review datasets, we have $\eta > 0.5$. Therefore, we cannot rely solely on the performance of individual views to determine the value of η . Generally, the better performing view does not automatically deserve a higher weightage. The Adaptive Two-view PA algorithm can solve this problem by adaptively updating η at each round of the learning process. This makes our adaptive Two-view PA algorithm both practical and robust.

Table 6.5: The Adaptive Two-view PA Results for All Datasets

Dataset	η	F-measure
Ads	0.342 ± 0.015	84.96 ± 2.41
Math Overflow	0.524 ± 0.001	75.70 ± 3.37
Product Review	0.795 ± 0.011	88.72 ± 1.80
WebKB	0.438 ± 0.001	98.02 ± 2.14

6.4 Summary

We proposed a Two-view Passive Aggressive algorithm, which is able to take advantage of multiple views of data to achieve an improvement in the overall classification performance. We formulate our learning framework into an optimization problem and derive a closed form solution. We subsequently enhanced the Two-view PA algorithm to obtain an adaptive version called the Adaptive Two-view PA algorithm, which has proved to be useful in practice.

Our closed form solutions do not need to employ convex optimization frameworks such as CVX² and CVXOPT³.

We evaluated the proposed approach on practical applications such as product review classification, advertising image removal, and math topic classification. We also prepared a two-view Math Overflow dataset containing text and math expressions, which is useful for math topic classification.

²<http://cvxr.com/cvx/>

³<http://abel.ee.ucla.edu/cvxopt/>

Chapter 7

Online Learning to Rank

In chapters 5 and 6, we have proposed large margin algorithms for classification. In this chapter, we propose a large margin online learning algorithm for ranking. The proposed algorithm was evaluated on math document dataset, which comprise two views of data such as math expression and textual data.

Math expressions are highly symbolic and structured. Searching math expressions is a challenging problem. For example, consider the following two math expressions: $(x + 1)$ and $e^{(x+1)}$. In each expression, the role of the sub-expression $(x + 1)$ is totally different. Therefore, if a query is to find expressions with the exponent $(x + 1)$, then all relevant expressions with $(x + 1)$ as its exponent should be ranked higher in the search result. In addition, math expressions also have semantic meanings. For example, the two expressions x^2 and y^2 should have the same semantic meaning, i.e., power of 2, despite the different variable names. In another example, consider the expressions $\int \sin(\log(x))dx$ and $\int \log(\sin(x))dx$. The order of the elements in the math expressions also indicates a certain semantic meaning. Any change in the order will also change the semantic meaning of the expression. When a query is issued for math search, it is important to consider the exact order of the elements in the query expression in order to rank the retrieved math expressions correctly.

In this chapter, we evaluate the proposed learning-to-rank algorithm on the math document data, which comprises two steps: (1) extract meaningful math index terms based on the semantically rich *MathML* format; (2) apply the algorithm to improve the accuracy of the ranking process.

Figure 7.1 shows the proposed mathd document search framework. The idea is to extract and index text and math terms independently in two separate inverted indexes. A user query is first matched against both the math and text inverted indexes to retrieve an initial set of

Figure 7.1: Math Document Search Framework

unranked candidates based on the intersection of matching documents in both the math and text query terms. A continuously learning *ranking model* assigns scores to each document in this candidate set.

7.1 Math Document Preprocessing

Math documents can be significantly more complex than textual documents; math expressions are sequence sensitive in two dimensions, e.g., matrices, fractions, nested forms. Conventional text retrieval methods, which typically throw away the linear sequence information in text, are thus extremely poorly suited to process math documents. Therefore, new retrieval methods must be devised to effectively handle math documents. We introduce our math document format and feature extraction approach in this section.

7.1.1 Math Document Representation

We choose $\text{\LaTeX}$ as a math markup language to store math documents. However, during the feature extraction step, we transform our $\text{\LaTeX}$ queries and documents into MathML, which is easier for machines to analyze. Listing 7.1 shows a $\text{\LaTeX}$ format sample comprising the title and content.

Listing 7.1: An Example Question

Formulate edge length problem as convex optimization problem

I want to use convex optimization to describe a problem in computational geometry. Let $E = (E_1, E_2, \dots, E_m)$ be a sequence of line segments in the plane, where E_1 and E_n may be points and the rest are non-degenerate segments. A *critical path* on E is a selection of points $(p_1, p_2, \dots, p_m)$ with $p_i \in E_i$ with edges $e_i = (p_i, p_{i+1})$ such that (1) all e_i has the same length l and (2) no other selection of points results in a path with edges that are not longer than l and some that are shorter...

7.1.2 Math Document Feature Extraction

Due to the existence of text and math data in math documents, the preprocessing step for math documents is more complicated than that of normal text documents. For the purpose of math search, we treat text and math data separately. While the text feature set is extracted using the English tokenizer, the math feature set is extracted using the math feature selection method described in Section 6.1 of Chapter 6.

7.2 Data Model

In this section, we define two types of index terms, namely text and math. These two types of index terms collectively form our math-aware search engine.

Definition 7.1 (Text Index Term). Let n be the number of index terms in the system. Denote each text index term by t_i , and the set $T = \{t_1, \dots, t_n\}$ contains all text index terms.

Definition 7.2 (Math Index Term). Let m be the number of math index terms in the system. Each math index term m_j is an element of the math feature set. Let $M = \{m_1, \dots, m_m\}$ be the set of all math index terms.

Let $\mathbb{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_k\}$ and $\mathbb{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l\}$ be a collection of k math documents and a set of l user queries, respectively. The math-aware information retrieval problem aims to determine a *ranking function* $\text{score} : \mathbb{D} \times \mathbb{Q} \mapsto \mathbb{R}$, which defines an ordering among documents based on the queries via a *framework* F . This framework considers all math documents and queries. For example, in text information retrieval, the framework F is typically the vector space of document and query points, with the ranking function determined by cosine similarity.

In this section, we formally define the query and ranking models for math Q&A search based on the vector space model. The proposed approach is an extension of the vector space model for conventional text information retrieval.

7.2.1 Matching Model

In this section, we formally define math documents and queries in the math-aware search engine, where the data consists of both text and math expressions. The search engine should index both kinds of data simultaneously.

Definition 7.3 (Math Predicate). A math predicate is defined as p_i if the math index term m_i appears in a document $\mathbf{d}_j$. In other words, $\mathbf{d}_j$ satisfies p_i if $\mathbf{d}_j$ contains the math index term m_i .

Definition 7.4 (Math Specification). A math specification $P = p_1 \wedge p_2 \wedge \dots \wedge p_s$ specifies a subset of documents $\mathbb{D}_P \subset \mathbb{D}$, where $\mathbf{d}_i \in \mathbb{D}_P$, if $\mathbf{d}_i$ satisfies P .

Definition 7.5 (Math Document). A math document $\mathbf{d}_j \in \mathbb{D}$ is determined by a vector $\mathbf{d}_j = (d_{1j}, \dots, d_{nj})$ and a math specification P_j , where $d_{ij} \geq 0$. If the text index term t_i appears in document $\mathbf{d}_j$, then $d_{ij} > 0$; otherwise, $d_{ij} = 0$. Thus P_j is the corresponding math specification satisfied by document $\mathbf{d}_j$.

Definition 7.6 (Math Query). A query in a math question answering system, $\mathbf{q} = \mathbf{q}_t | \mathbf{q}_m$, is comprised of two parts: a keyword query $\mathbf{q}_t \subset T$ and a math expression query $\mathbf{q}_m \subset M$. The keyword query $\mathbf{q}_t$ is a list of keywords and the math expression query $\mathbf{q}_m$ is a list of math expressions.

For the math query $\mathbf{q} = \mathbf{q}_t | \mathbf{q}_m$, suppose that $\mathbf{q}_t = \{t_1, \dots, t_i\}$ and $\mathbf{q}_m = \{m_1, \dots, m_j\}$ correspond to math specification P , the unranked result $\mathbf{q}(\mathbb{D})$ denotes the subset of documents that satisfies P , and those that contain all the terms in $\mathbf{q}_t$ as

$$\mathbf{q}(\mathbb{D}) = \sigma_P(\mathbb{D}) \cap \sigma_{t_1}(\mathbb{D}) \cap \dots \cap \sigma_{t_i}(\mathbb{D})$$

where the selection function $\sigma_{t_i}(\mathbb{D}) = \{\mathbf{d}_j | t_i \in \mathbf{d}_j \wedge \mathbf{d}_j \in \mathbb{D}\}$.

7.2.2 Ranking Model

A Straightforward Approach

Now, we define the ranking of the result of $\mathbf{q}$. We start by presenting a generic representation of the conventional ranking function for keyword queries. We then extend it to a math document ranking function.

Given a query $\mathbf{q}$ and a document $\mathbf{d} \in \mathbb{D}$, a conventional ranking function $f(\cdot)$ takes as argument statistics from $\mathbf{q}$, $\mathbf{d}$, and computes a score of $\mathbf{d}$ with respect to $\mathbf{q}$ as

$$\text{score}(\mathbf{q}, \mathbf{d}) = f(\mathbf{q}_t, \mathbf{q}_m, \mathbf{d})$$

For example, $tf \times idf$ weighting [62, 82] is a well-known ranking model. Among its variants, the pivoted normalization formula is considered as one of the best performing vector space models, and is widely used in many text search systems as

$$\text{score}(\mathbf{q}, \mathbf{d}) = \sum_{t \in \mathbf{q}} \frac{1 + \ln(1 + tf(t, \mathbf{d}))}{(1 - s) + s \cdot \frac{\text{len}(\mathbf{d})}{|\mathbb{D}|}} \cdot tf(t, \mathbf{q}) \cdot \ln \frac{|\mathbb{D}| + 1}{df(t, \mathbb{D})}$$

where $\text{len}(\mathbf{d})$ is the length of document $\mathbf{d}$, $|\cdot|$ denotes the set cardinality function, $tf(t, \mathbf{d})$ is the term frequency of term t in document $\mathbf{d}$, $df(t, \mathbb{D})$ is the document frequency of term t in the document collection $\mathbb{D}$, and s is a parameter.

We note that the scoring function depends on the number of documents. Therefore, we revise this scoring function for ranking math document search results as follows.

$$\text{score}(\mathbf{q}, \mathbf{d}) = \sum_{t \in \mathbf{q}} \frac{1 + \ln(1 + tf(t, \mathbf{d}))}{(1 - s) + s \cdot \frac{\text{len}(\mathbf{d})}{|\mathbb{D}_P|}} \cdot tf(t, \mathbf{q}) \cdot \ln \frac{|\mathbb{D}_P| + 1}{df(t, \mathbb{D}_P)} \quad (7.1)$$

In this scoring formula, we replaced $\mathbb{D}$ by $\mathbb{D}_P$. The revised scoring function does not depend on the whole collection of documents but instead on a subset of documents $\mathbb{D}_P$.

Ranking Passive Aggressive Algorithm

In this section, we describe the Ranking Passive Aggressive algorithm in details based on the Passive Aggressive algorithm [21]. Let $(\mathbf{x}_t, y_t)_{t=1, T}$ be a sequence of examples, Crammer et al. proposed the PA algorithm, which learns the weight $\mathbf{w}$ of a linear prediction function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ based on the following optimization problem,

$$\begin{aligned} \mathbf{w}_{t+1} = & \underset{\mathbf{w} \in \mathbb{R}^n}{\text{argmin}} \quad \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 \\ \text{s.t.} \quad & \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0 \end{aligned} \quad (7.2)$$

where ℓ is the Hinge loss function.

We adapt the Passive Aggressive algorithm to learn-to-rank by training on pairs of instances. A correct classification corresponds to ranking a pair so that the more relevant instance is scored higher and vice versa.

We know that math and text features are extracted from different data types and therefore should be treated differently. We can apply the cascade ranking model [85], where one type of data is used to rank documents at each stage of a multi-stage ranking procedure. However, in practice, this approach may not be practical because math questions do not always contain math expressions. Therefore, we propose another approach that combines math and text data together as follows. Let $\mathbf{a}_t \in \mathbb{R}^k$ and $\mathbf{b}_t \in \mathbb{R}^l$ be the math and text vectors representing the document $\mathbf{d}_t \in \mathbb{R}^n$ respectively, where $k + l = n$. The simplest way is to concatenate math and text data as

$$\mathbf{d}_t = (\eta \mathbf{a}_t, (1 - \eta) \mathbf{b}_t)$$

where $\eta \in [0, 1]$ weighs the importance between the text and math data.

Remark $\eta = 0$ means that we remove math expressions from the math document. If $\eta = 1$, we only rank the search result based on the math expressions and ignore text data. Normally, η is assigned a value between 0 and 1. If we treat math and text data equally, then $\eta = 0.5$. If $\eta < 0.5$, we consider text data to be more important than math data, and vice versa.

Let the feature score vector of a document $\mathbf{d}_t \in \mathbb{R}^n$ and query $\mathbf{q}$ be defined as follows.

$$\phi_t = (f_1(\mathbf{d}_t, \mathbf{q}), \dots, f_n(\mathbf{d}_t, \mathbf{q}))$$

The goal is to learn the weight vector $\mathbf{w}$ of the following scoring function.

$$\text{score}(\phi_t, \mathbf{w}) = \mathbf{w} \cdot \phi_t \quad (7.3)$$

Suppose that R and N denote relevant and non-relevant feature score vectors. Then the feature score vectors of relevant and non-relevant documents to the query $\mathbf{q}$ are defined as ϕ_t^R and ϕ_t^N , respectively. Let $(\phi_t^R, \phi_t^N)_{t=1, T}$ be a sequence of training examples. We should have $\mathbf{w} \cdot \phi_t^R \geq \mathbf{w} \cdot \phi_t^N$ because the score of the relevant document to the query should be greater than that of the non-relevant document. For the Ranking Perceptron [33], the weight vector $\mathbf{w}_{t+1}$ is learnt on each round t as,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau(\phi_t^R - \phi_t^N)$$

where τ is a constant learning rate. Similar to the Perceptron algorithm, the Ranking Perceptron algorithm is very sensitive to the learning rate, and works poorly on non-linearly separable data.

To overcome this problem, we introduce a new Hinge loss function

$$\ell((\phi_t^R, \phi_t^N); \mathbf{w}) = \max \left\{ 0, 1 - (\mathbf{w} \cdot \phi_t^R - \mathbf{w} \cdot \phi_t^N) \right\}$$

which penalizes the difference between the two scores if the score of the non-relevant document is greater than that of the relevant document, with respect to the query. We then derive a soft margin optimization problem by introducing a slack variable ξ as

$$\begin{aligned} \mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathbb{R}^n \wedge \mathbf{w} \succeq 0}{\operatorname{argmin}} \quad & \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi \\ \text{s.t.} \quad & 1 - (\mathbf{w} \cdot \phi_t^R - \mathbf{w} \cdot \phi_t^N) \leq \xi; \quad \xi \geq 0 \end{aligned} \quad (7.4)$$

where C is an aggressiveness parameter.

Proposition 7.1. The optimization problem (7.4) has the closed-form solution as

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t(\phi_t^R - \phi_t^N)$$

where the learning rate τ_t has the form

$$\tau_t = \min \left\{ C, \frac{1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)}{(\phi_t^R - \phi_t^N)^2} \right\}.$$

Algorithm 7.3 summarizes the *Ranking Passive Aggressive* algorithm.

Algorithm 7.3 Ranking Passive Aggressive Algorithm

Input:

C = positive aggressiveness parameter

Output:

None

Process:

Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$;

for $t = 1, 2, \dots$ **do**

Receive feature score vectors $\phi_t^R \in \mathbb{R}^n$ and $\phi_t^N \in \mathbb{R}^n$

Suffer loss $\ell_t \leftarrow \max \left\{ 0, 1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N) \right\}$

if $\ell_t > 0$ **then**

Set $\tau_t \leftarrow \min \left\{ C, \frac{1 - (\mathbf{w}_t \cdot \phi_t^R - \mathbf{w}_t \cdot \phi_t^N)}{(\phi_t^R - \phi_t^N)^2} \right\}$

Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \tau_t(\phi_t^R - \phi_t^N)$

end if

end for

Let ϕ_t^R and ϕ_t^N be the feature score vectors of the relevant and non-relevant documents for query $\mathbf{q}$, respectively. Let $\mathbf{w}_t$ be the weight vector of the Ranking Passive Aggressive algorithm on round t . We say that the proposed algorithm makes a mistake if $\mathbf{w}_t \cdot \phi_t^R < \mathbf{w}_t \cdot \phi_t^N$. The total number of mistakes made by the proposed algorithm is bounded as shown in Proposition 7.2.

Proposition 7.2. Let $(\phi_t^R, \phi_t^N)_{t=1,T}$ be a sequence of examples, where $\phi_t^R \in \mathbb{R}^n$ and $\phi_t^N \in \mathbb{R}^n$ for all t . Then for any weight vector $\mathbf{u} \in \mathbb{R}^n$ and its loss $\ell_t^* = \ell((\phi_t^R, \phi_t^N); \mathbf{u})$, the number of mistakes made by the Ranking Passive Aggressive algorithm is bounded by

$$\max \left\{ \frac{1}{C}, \max_{t=1,T} (\phi_t^R - \phi_t^N)^2 \right\} \left(\|\mathbf{u}\|^2 + 2C \sum_{t=1}^T \ell_t^* \right)$$

Adding Metadata to Math Question Retrieval

In math question answering system, math questions typically contain additional metadata such as user ratings, number of views, number of answers, etc. The metadata can be very

useful for improving ranking results, faceted search, and personalization [5]. The question is how to combine the document content and its metadata to answer a user's information need. To address this question, we have to first choose a suitable set of metadata. For instance, a user may prefer to see the most popular and/or answered questions. Moreover, he could also be looking for questions that have already been answered by other users. In each of these two scenarios, the number of views and answers are useful information for re-ranking the search results.

To retrieve documents based on user rating, Zhang et al. [91] proposed a simple information retrieval technique, which considers thumbs-up and thumbs-down as a term in the document. They came up with a scoring function, which is the ratio of the number of thumbs-up (n^+) to the number of thumbs-down (n^-). However, practical systems like Math Overflow only store the difference ($n^+ - n^-$) between these two user ratings. Therefore, the above technique cannot be applied directly to these systems. In order to overcome this limitation and take other kinds of metadata into account, we propose another information retrieval approach, which is based on the $tf \times idf$ ranking model.

Since users want to retrieve “good” math questions, e.g., questions with thumbs-up, questions with many views, questions with several answers, questions with highly-rated answers, and questions with high reputation, we can embed this need into a virtual query $\mathbf{v}$ of metadata. The scoring function of this query is defined as

$$score(\mathbf{d}_t, \mathbf{v}) = \sum_{w \in \mathbf{v}} \frac{1 + \ln(1 + \mathbf{m}_t(w))}{(1 - s) - s \frac{\ln(\mathbf{m}_t)}{\text{avg}(\ln(\mathbf{m}_t))}} \quad (7.5)$$

where $\mathbf{m}_t$ is the metadata of document $\mathbf{d}_t$.

Let the scoring function between document $\mathbf{d}_t$ and query $\mathbf{q}$ be defined in Equation (7.3). The overall scoring function is defined as the linear combination of the query and virtual-query scoring functions, $score(\mathbf{d}_t, \mathbf{q})$ and $score(\mathbf{d}_t, \mathbf{v})$,

$$score(\mathbf{d}_t, \mathbf{q}, \mathbf{v}) = \alpha \cdot score(\mathbf{d}_t, \mathbf{q}) + (1 - \alpha) \cdot score(\mathbf{d}_t, \mathbf{v}) \quad (7.6)$$

where $\alpha \in [0, 1]$ determines the relative importance of content and metadata for document $\mathbf{d}_t$. The advantage of this approach is that we are able to determine the metadata score beforehand and update it whenever users update the metadata.

7.3 Performance Evaluation

7.3.1 Dataset

The major obstacle to evaluating math search is that we do not have a standard benchmark dataset like for other more common information retrieval tasks. Also, it is hard to compare our proposed approach with other systems such as MathFind and Wolfram Function Search because they are either unavailable or inaccessible.

Therefore, we build our own math search dataset by crawling and downloading 31,288 math questions and answers (or math documents for short) from the online Math Overflow question answering system. Each math document can have one or more of the 20 categories from the system. We parsed and converted the math documents from HTML into the *Text REtrieval Conference (TREC)*¹ XML format. We also determined and stored 30 queries in the TREC format. Each query has a set of relevant and non-relevant documents. The relationship (relevant and non-relevant) between the documents and queries is defined in another file, also stored in the TREC format.

7.3.2 Experimental Setup

Each document is identified by a numerical document ID. All math documents are indexed using inverted index techniques with stop-word removal based on the English stop-word list provided by *Terrier Information Retrieval Platform* [71]. The total number of index terms is 29,745.

To learn the ranking function, all math documents and queries are processed and converted into $tf \times idf$ [62] vectors. The feature score vectors are calculated based on these documents and queries. The feature score vectors are used to train the Ranking Perceptron and Ranking PA algorithm.

After training, we obtained a weight vector $\mathbf{w}$, which is used to determine the scores of all retrieved math documents. Since other math-aware search systems only focus on math feature extraction and apply traditional information retrieval techniques, we only compared our approach against standard retrieval methods such as BM25 [44], InL2 [36], and $tf \times idf$, where InL2 stands for “Inverse Document Frequency with Laplace after-effect and normalization 2”. While BM25 and $tf \times idf$ are established information retrieval techniques, the InL2 model was only proposed recently. For the $tf \times idf$ ranking model, we applied the straightforward approach described in Section 7.2.2.

¹<http://trec.nist.gov/>

7.3.3 Math Feature Extraction Evaluation

In this section, we evaluate the proposed math feature extraction approach by comparing the retrieval precision on the math expressions with and without math feature extraction. The procedure is as follows:

- Extract all math expressions from the Math Overflow dataset to create a math-only dataset, called Math Expression dataset.
- Generate a textual math feature set, where the math expressions are treated as normal text.
- Generate a math feature set, where math features are extracted as mentioned in Section 6.1 in Chapter 6.
- Generate an n-gram feature set as mentioned in [67]
- Conduct the experiments on these three feature sets and compare their performance.

Figure 7.2: FSM for Extracting Textual Math Features

A straightforward approach to textual math feature extraction is to utilize a finite state machine (FSM) as shown in Figure 7.3, where whitespace stands for invisible characters, end of stream characters, and delimiters such as space, tab, comma, etc. Operator stands for common math operators such as plus, minus, power, etc. Literals include the characters from the set ‘a’ to ‘z’ and ‘A’ to ‘Z’. Digits refer to numbers ‘0’ to ‘9’. The state machine accepts a math expression as a sequence of characters and extracts math operators, math variables,

Table 7.1: Textual Math and Math Feature Properties

	Textual Math	Math Feature	N-Gram
# Unique Terms	2959	6060	7145
Avg. DocLen	40.25	13.46	65.55
Avg. PLen	25.53	8.63	28.01
Avg. QT (ms)	62.20	53.80	78.80

function names, and constant numbers. Specifically, at state 2, it skips a whitespace. At state 3, it extracts the operator. At termination, it returns a variable, a function name, or a constant number at state 4.

To compare the three feature sets, we show their properties in Table 7.1. We counted the number of unique terms and calculated the average *document length* (DocLen) and average *posting list length* (PLen), where the document length is determined based on the number of index terms in the document (i.e., math expressions). While the number of unique terms represents the size of the *lexicon*, the average posting list length indicates the number of documents associated with a term in the inverted index. These two major factors impact the performance of a search system in terms of *query time* (QT) and accuracy.

We note that the total number of unique terms in the math feature set is greater than that in the textual math feature set. However, the average document length and the average posting list length of the math feature set is smaller than those of the textual math feature set. This implies that the math feature set is less uniform and therefore more discriminative than the textual math feature set.

Figure 7.3: Performance Comparison on the Math Expression Dataset

To verify the above claim, we carried out a retrieval experiment and compared the IR performance on both feature sets. Figure 7.4 shows the precision at different recall values on the three feature sets. In this experiment, the popular ranking method BM25 was applied.

We see that BM25 does not work well on the textual math dataset. Its precision on this set is at most 6% at 40% recall. On the other hand, its precision on the math feature set and n-gram is significantly larger. Even in the worst case, the math feature set’s precision is improved by more than 53%. This result confirms that math feature extraction is essential for an effective math-aware search engine, compared to the simplistic textual math set. Besides, the math feature set is better than n-gram except at 10% recall.

7.3.4 Ranking Algorithm Evaluation

We trained both the Ranking Perceptron and Ranking PA algorithms on the feature score vectors. We then compared their performance by using the standard cumulative error rate, which is the ratio of mistakes made by the online learning algorithm over the total number of examples observed to-date. In this experiment, the performance of the Ranking PA algorithm and the Ranking Perceptron algorithm [33] is evaluated on the Math Overflow dataset.

Figure 7.4: Cumulative Error Rate on the Math Overflow Dataset

Figure 7.5 shows the cumulative error rates of both Ranking Perceptron and Ranking PA algorithms on the Math Overflow dataset. Compared to the Ranking Perceptron algorithm, the proposed Ranking PA algorithm consistently achieves a lower cumulative error rate than the Ranking Perceptron algorithm. This shows that the proposed algorithm is effective for improving the online ranking function learning. Hence, in the ranking process, if the weight of the Ranking PA is utilized, the retrieval precision will be improved compared to that of the Ranking Perceptron algorithm. Additional retrieval results in the next section will confirm this claim.

Table 7.2: Metadata Properties (* using Equation (7.5)).

	Rate	#Ans	#Views	Ans Rate	Score*
Min	-3	0	14	-4	2.9215
Max	93	121	21606	86	27.2277
Avg	5.9049	1.9146	498.2342	5.8395	14.8662

7.3.5 Retrieval Performance Evaluation

For the Ranking PA algorithm, we used cross validation to choose the optimal parameter C . Then, we used the optimal solution/weight to rank retrieved documents based on the scoring function in Equation (7.6) in Section 7.2.2. We also evaluated the Ranking Perceptron algorithm. In this experiment, we used both text and math feature sets, where the text features were extracted using an English tokenizer with stop-word removal and the math features were extracted as mentioned in Section 6.1 in Chapter 6. Moreover, the metadata of each question is extracted to calculate the score. We found that the optimal value of α in Equation (7.6) to be around 0.958. This means that content largely rules over metadata. The properties of the metadata of all questions are shown in Table 7.2.

Figure 7.6 shows the performance comparison between our proposed approach and other methods. The retrieval precision of each method is compared at recall levels of 0%, 10%, and 20%. The experimental results show that the Ranking PA algorithm consistently outperforms other methods. Its precision is at least 9% better than other methods at 0% and 10% recall levels. And in the worst case, the improvement is more than 7% at 20% recall.

Figure 7.5: Performance Comparison on the Math Overflow Dataset

Compared with other methods, the straightforward approach $tf \times idf$ is slightly better. The improvement is nearly 1%. Hence, term weighting method alone is not good enough

to rank highly structured data such as math expressions. In this case, the learning-to-rank algorithm should be applied to improve the performance of the search system. In this experiment, the Ranking PA has been shown to be an elegant solution for this problem. If we take into account the metadata, the performance is just marginally improved. This improvement is nearly 5% at 20% recall but declines gradually with decreasing recall levels.

7.4 Summary

In this chapter, we proposed an online learning-to-rank algorithm and a math search framework. The search framework consists of two major steps: feature extraction and learning-to-rank. In feature extraction, we used the method for math expression feature extraction based on the content MathML format. To learn the score function, we formulated and derived an online optimization problem. We then derived a closed-form solution, leading to our Ranking PA algorithm. In addition, we derived the mistake bound of the Ranking PA algorithm.

Moreover, we crawled the Math Overflow question answering system and generated a new dataset for math document retrieval. Our Math Overflow dataset is a big dataset, which has more than 30,000 math documents. It is very useful for math document retrieval evaluation in general, and we plan to release it publicly for the benefit of other researchers in math search. We evaluated our proposed ranking approach and benchmarked it against other baseline techniques on the Math Overflow dataset. The experimental results show that the proposed approach performs better than the runner-up by more than 9%. Although we evaluated our proposed approach on the math question answering dataset, it is possible to apply it to other kinds of math document retrieval.

Chapter 8

Conclusion and Future Work

8.1 Summary of the Thesis

The summary of this thesis is shown in Figure 8.1, where we focused on online learning to solve two major problems in search and classification. In both problems, machine learning approaches were used to improve the performance.

Figure 8.1: Thesis Summary

The research outcome includes improved supervised term weighting methods, a centroid-based classification algorithm, new two-view online learning classification algorithms, and an online learning-to-rank algorithm geared for math search. Specifically, the main contribution are listed as follows.

- Supervised term weighting methods. The research has a number of important implications for future practice. First, by taking advantage of labeled data, we are able to improve the performance of classifiers. For binary text categorization, experimental results show more than 10% improvements. Second, the proposed supervised term weighting methods work very well on imbalanced datasets compared to existing approaches. For multi-class problems, the proposed $tf \times JS$ term weighting can effectively deal with imbalanced data while boosting the weights of class-specific terms.
- Centroid-based classifiers for text categorization. Taken together, the investigation of supervised term weighting methods and centroid-based classifiers suggest that these classifiers are more practical in terms of both learning speed and classification accuracy. While their performance is comparable to the state-of-the-art SVM, their learning is linear with respect to the training dataset size. They not only are faster than SVM in learning but also have an incremental learning capability. However, since our proposed centroid-based classification algorithm utilized the supervised term weighting approach, centroid classifiers works well only for two-class datasets in practice. That is also the limitation of the supervised term weighting approach. Experimental results on real world datasets shows that centroid-based classifiers with supervised term weighting performed 15% better than the best centroid-based classification method (with unsupervised term weighting).
- Online learning. In this investigation, the aim was to develop online learning algorithms for text categorization, math topic classification, and search result re-ranking. In order to improve the performance of the current online learning algorithm, a distribution-aware passive aggressive algorithm was proposed. This algorithm takes into account the distribution of data around the decision hyperplane. To classify math topics, we proposed an online two-view learning algorithm, which can work on both text and math data at the same time. Furthermore, in an attempt to improve the accuracy of the math search system, an online learning-to-rank algorithm has been proposed. In this approach, the scoring function between the documents and queries is learnt from the training data, where a sequence of relevant and irrelevant document pairs with respect to a query was determined manually.

8.2 Future Work

The work reported in this thesis is just a small first step towards applying machine learning approaches combined with supervised term weighting methods to information retrieval. Information retrieval approaches should take into account other kinds of data of the documents to improve the performance of search systems. For example, documents collected from social networks such as question and answering systems, online forums, etc., typically contain valuable metadata such as user profiles, user ratings, number of views, which are very useful in the ranking process.

8.2.1 Supervised Term Weighting

Theoretical bounds on the improvement in squared loss of supervised term weighting schemes such as $tf \times KL$ and $tf \times JS$ compared to the baseline $tf \times idf$ can be derived. More investigations are also needed to qualify and quantify the impact of generalized $tf \times KL$ on the negative class performance for highly skewed datasets, since we have observed some anomalies in $tf \times KL$ performance there. Performance of CF-JS and SVM-JS on true multi-class datasets could be studied to evaluate the impact of supervised term weighting in a multi-class setting.

8.2.2 Online Multi-class Learning

For online learning, we are currently extending the PAM family of algorithms for multi-class and structural data problems. We are also refining the analytical error loss bounds of PAM, so as to stipulate the data conditions for which PAM will be decisively superior and vice-versa. For future work, we would also want to evaluate extensively the practical performance of AROW versus PAM-II, given their similarities in the weight update equations. In particular, we want to find out if the adaptive update of the covariance matrix in AROW is superior to PAM-II's static covariance update.

Our proposed online learning algorithms have been shown to outperform other algorithms. However, they are fundamentally binary classification algorithms. Although we can apply the binary classification algorithms for multi-class problem by using one-against-all, one-against-one, and discriminant functions approaches, the problem is still not fundamentally solved.

For future work, we will focus on extending the online learning process to multi-class problem and apply it to *math symbol recognition*. To solve it, we can generalize the binary

classification problem using the approach proposed by Collins [17]. Given an input space $\mathbb{X}$ taking labels from a finite set $\mathcal{Y}$, these two problems are defined as follows. Assuming that there is a feature function $f(\mathbf{x}, y) \in \mathbb{R}^d$ that maps instances $\mathbf{x} \in \mathbb{X}$ and labels $y \in \mathcal{Y}$ into a common space. The problem is to learn the linear model parameter $\mathbf{w}$ of the prediction function, which has the form:

$$\hat{y} = \operatorname{argmax}_{z \in \mathcal{Y}} \quad \mathbf{w} \cdot f(\mathbf{x}, z)$$

Lastly, extending online classification algorithms over a family of metrics other than the Euclidean and Mahalanobis metric deserves further examination [65, 28].

8.2.3 Math Topic Classification

Although we have evaluated our proposed approach on math questions and answers, it can be applied in practice to deal with other kinds of math documents such as math questions in student books, scientific papers, etc. There remain some interesting open problems that warrant further investigations. First, the math feature extraction method should be investigated further because the performance based on math features only is not good enough. Since math feature extraction is not mature, we were managed to include text features to improve the performance of math topic classifiers. Then, we would also like to extend the Two-view PA algorithm to handle multiple views and multiple classes. However, formulating a multi-view PA is non-trivial, as it involves defining multi-view relatedness and minimizing pairs of view agreements. Formulating a multi-class Two-view PA should be more feasible.

Moreover, we are extending our research work by integrating math search with text retrieval for mathematical document retrieval and visualization. As such, scientific math documents can be indexed, retrieved, and visualized using both textual data and math expression.

Appendix A

List of Publications

The work reported in this thesis has been published/accepted at the following international journals, conferences, or workshops.

A.1 Journal Papers

1. Tam T. Nguyen, Siu Cheung Hui, Kuiyu Chang. (2011). A lattice-based Approach for Mathematical Search using Formal Concept Analysis. *Expert Systems with Applications*, 39(5): 5820-5828, April, 2012.
2. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2012). Supervised Term Weighting Centroid-based Classifiers for Text Categorization. **Invited Paper**. *Knowledge and Information Systems (KAIS)*, pages 1-25.

A.2 Conference and Workshop Papers

1. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2012). A Math-Aware Search Engine for Math Question Answering System. *ACM International Conference on Information and Knowledge Management (CIKM)*.
2. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2012). Adaptive Two-View Online Learning for Math Topic Classification. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*.
3. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2012). Two-View Online Learning. *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (**Best Paper Award**).

4. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2011). Word Cloud Model for Text Categorization. *IEEE International Conference on Data Mining (ICDM)*.
5. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2011). Distribution-Aware Online Classifiers. *International Joint Conferences on Artificial Intelligence (IJCAI)*.
6. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2011). Supervised Term Weighting for Sentiment Analysis. *IEEE International Conference on Intelligence and Security Informatics (ISI)*.
7. Tam T. Nguyen, Kuiyu Chang, Siu-Cheung Hui. (2011). Probabilistic Supervised Term Weighting for Binary Text Categorization. *The 3rd Workshop on Social Web Search and Mining (SWSM) in conjunction with SIGIR 2011*.

Bibliography

- [1] S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society*, pages 131–142, 1966.
- [2] H. Altincay and Z. Erenel. Analytical evaluation of term weighting schemes for text categorization. *Pattern Recognition Letter*, 31:1310–1323, August 2010.
- [3] A. Asuncion and D. Newman. Uci machine learning repository, 2007.
- [4] R. Bekkerman and M. Gavish. High-precision phrase-based document classification on a modern scale. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 231–239, New York, NY, USA, 2011. ACM.
- [5] P. N. Bennett, K. El-Arini, T. Joachims, and K. M. Svore. Sigir '11 workshop report: Enriching information retrieval, 2011.
- [6] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *The Association for Computer Linguistics (ACL)*, 2007.
- [7] H. Block. The perceptron: A model for brain functioning. *Rev. Modern Phys.*, 34:123–135, 1962.
- [8] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT'98, pages 92–100, New York, NY, USA, 1998. ACM.
- [9] A. Bordes and L. Bottou. The huller: a simple and efficient online svm. *Machine Learning: ECML 2005*, pages 505–512, 2005.

- [10] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with larank. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 89–96, New York, NY, USA, 2007. ACM.
- [11] A. Bordes, N. Usunier, and L. Bottou. Sequence labelling svms trained in one pass. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I, ECML PKDD '08*, pages 146–161, Berlin, Heidelberg, 2008. Springer-Verlag.
- [12] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory, COLT '92*, pages 144–152, New York, NY, USA, 1992. ACM.
- [13] R. F. Bruce and J. M. Wiebe. Recognizing subjectivity: a case study in manual tagging. *Natural Language Engineering*, 5:187–205, 1999.
- [14] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06*, pages 186–193, New York, NY, USA, 2006. ACM.
- [15] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34, 2005.
- [16] 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>.
- [17] M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA, 2002. Association for Computational Linguistics.
- [18] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, volume 20, pages 273–297. Springer Netherlands, 1995.
- [19] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, Sept. 2006.
- [20] J. Cramer. The origins of logistic regression. Tinbergen Institute Discussion Papers 02-119/4, Tinbergen Institute, Dec. 2002.

- [21] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, pages 551–585, 2006.
- [22] K. Crammer, M. Dredze, and A. Kulesza. Multi-class confidence weighted algorithms. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 496–504, Singapore, August 2009. Association for Computational Linguistics.
- [23] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems*, 2009.
- [24] F. Debole and F. Sebastiani. Supervised term weighting for automated text categorization. In *Proceedings of the 2003 ACM symposium on Applied computing*, SAC '03, pages 784–788, New York, NY, USA, 2003. ACM.
- [25] S. Dey. A multi-classifier system for text categorization. In *Proceedings of the 2011 ACM Symposium on Research in Applied Computation*, RACS '11, pages 325–329, New York, NY, USA, 2011. ACM.
- [26] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 264–271, New York, NY, USA, 2008. ACM.
- [27] H. Drucker, D. Wu, and V. N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.
- [28] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT '10: Proceedings of the 23rd Annual Conference on Learning Theory*, 2010.
- [29] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 256–263, New York, NY, USA, 2000. ACM.
- [30] A. Esuli and F. Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06)*, pages 417–422, 2006.
- [31] J. D. R. Farquhar, D. R. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szepesvári. Two view learning: Svm-2k, theory and practice. In *Proceedings of NIPS'05*, 2005.

- [32] W. B. Frakes and R. A. Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 1992.
- [33] J. Gao, H. Qi, X. Xia, and J.-Y. Nie. Linear discriminant model for information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 290–297, New York, NY, USA, 2005. ACM.
- [34] M. Géry and C. Largeron. BM25t: a BM25 extension for focused information retrieval. *Knowledge and Information Systems*, pages 1–25, June 2011.
- [35] H. Guan, J. Zhou, and M. Guo. A class-feature-centroid classifier for text categorization. In *18th International World Wide Web Conference*, pages 201–201, April 2009.
- [36] R. Guillén. Gir with language modeling and dfr using terrier. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, CLEF'08, pages 822–829, Berlin, Heidelberg, 2009. Springer-Verlag.
- [37] E.-H. Han and G. Karypis. Centroid-based document classification: Analysis and experimental results. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 424–431, London, UK, 2000. Springer-Verlag.
- [38] V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181, Morristown, NJ, USA, 1997. Association for Computational Linguistics.
- [39] V. Hatzivassiloglou and J. M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th conference on Computational linguistics*, pages 299–305, Morristown, NJ, USA, 2000. Association for Computational Linguistics.
- [40] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA, 2004. ACM.
- [41] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Machine Learning: ECML-98*,

- volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0026683.
- [42] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [43] T. Joachims. A statistical learning model of text classification with support vector machines. In *Proceedings of ACM SIGIR*, pages 128–136, 2001.
- [44] K. S. Jones. Index term weighting. *Information Storage and Retrieval*, 9(11):619 – 633, 1973.
- [45] K. N. Junejo and A. Karim. A robust discriminative term weighting based linear discriminant method for text classification. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 323–332, Washington, DC, USA, 2008. IEEE Computer Society.
- [46] K. N. Junejo and A. Karim. Robust personalizable spam filtering via local and global discrimination modeling. *Knowledge and Information Systems*, pages 1–36, Feb. 2012.
- [47] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 440–447, New York, NY, USA, 2008. ACM.
- [48] L.-W. Ku, L.-Y. Lee, T.-H. Wu, and H.-H. Chen. Major topic detection and its application to opinion summarization. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 627–628, New York, NY, USA, 2005. ACM.
- [49] O. Kurland and L. Lee. Clusters, language models, and ad hoc information retrieval. *ACM Transaction on Information System*, 27(3):1–39, 2009.
- [50] N. Kushmerick. Learning to remove internet advertisements. In *Proceedings of the third annual conference on Autonomous Agents*, AGENTS '99, pages 175–181, New York, NY, USA, 1999. ACM.
- [51] M. Lan, C. L. Tan, J. Su, and Y. Lu. Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 31:721–735, 2009.

- [52] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *AAAI'92: Proceedings of the tenth national conference on Artificial intelligence*, pages 223–228. AAAI Press, 1992.
- [53] N. Lavesson, M. Boldt, P. Davidsson, and A. Jacobsson. Learning to detect spyware using end user license agreements. *Knowl. Inf. Syst.*, 26(2):285–307, Feb. 2011.
- [54] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, London, UK, 1998. Springer-Verlag.
- [55] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *In Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.
- [56] C. H. Li, J. C. Yang, and S. C. Park. Text categorization algorithms using semantic approaches, corpus-based thesaurus and wordnet. *Expert Systems with Applications*, 39(1):765–772, Jan. 2012.
- [57] G. Li, S. C. H. Hoi, K. Chang, and R. Jain. Micro-blogging sentiment detection by collaborative online learning. In *Proceedings of the IEEE International Conference on Data Mining*, pages 893–898, Sydney, Australia, 2010.
- [58] S. Li, R. Xia, C. Zong, and C.-R. Huang. A framework of feature selection methods for text categorization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL-IJCNLP '09, pages 692–700, Morristown, NJ, USA, 2009. Association for Computational Linguistics.
- [59] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37:145–151, 1991.
- [60] W. Liu and T. Wang. Online active multi-field learning for efficient email spam filtering. *Knowledge and Information Systems*, pages 1–20, Nov. 2011.
- [61] J. Ma, A. Kulesza, M. Dredze, K. Crammer, L. K. Saul, and F. Pereira. Exploiting feature covariance in high-dimensional online learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 393–500, Sardinia, Italy, 2010.

- [62] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [63] J. Martineau, T. Finin, A. Joshi, and S. Patel. Improving binary classification on text problems using differential word features. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 2019–2024, New York, NY, USA, 2009. ACM.
- [64] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Champman and Hall/CRC, 2000.
- [65] H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *COLT '10: Proceedings of the 23rd Annual Conference on Learning Theory*, 2010.
- [66] B. R. Miller and A. Youssef. Technical aspects of the digital library of mathematical functions. In *Annals of Mathematics and Artificial Intelligence*, volume 38, pages 121–136, 2003.
- [67] R. Miner and R. Munavalli. An approach to mathematical search through query formulation and data normalization. In *Calculement '07 / MKM '07: Proceedings of the 14th symposium on Towards Mechanized Mathematical Assistants*, pages 342–355, Berlin, Heidelberg, 2007. Springer-Verlag.
- [68] A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.
- [69] T. T. Nguyen, K. Chang, and S. C. Hui. Word cloud model for text categorization. In *Proceedings of the 11th IEEE International Conference on Data Mining*, pages 487–496, 2011.
- [70] A. Novikoff. On convergence proofs of perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 7, pages 615–622, 1962.
- [71] I. Ounis, G. Amati, V. Plachouras, B. He, C. MacDonald, and D. Johnson. Terrier information retrieval platform. *Advances in Information Retrieval*, 3408:517–519, 2005.
- [72] B. Pang and L. Lee. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting*

- on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.
- [73] B. Pang and L. Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
- [74] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 79–86, Morristown, NJ, USA, 2002. Association for Computational Linguistics.
- [75] D. Puppin, F. Silvestri, R. Perego, and R. Baeza-Yates. Tuning the capacity of search engines: Load-driven routing and incremental caching to reduce and balance the load. *ACM Transation on Information System*, 28(2):1–36, 2010.
- [76] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80(3):227–248, 1989.
- [77] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.
- [78] G. Salton. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
- [79] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.
- [80] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.
- [81] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 824–831, New York, NY, USA, 2005. ACM.
- [82] A. Singhal. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24:2001, 2001.
- [83] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

- [84] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.
- [85] L. Wang, J. Lin, and D. Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 105–114, New York, NY, USA, 2011. ACM.
- [86] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
- [87] A. S. Youssef. Roles of math search in mathematics. In J. M. Borwein and W. M. Farmer, editors, *MKM '06: Proceedings of the 5th International Conference on Mathematical Knowledge Management*, pages 2–16, Berlin Heidelberg, 2006. Springer-Verlag.
- [88] H. Yu and V. Hatzivassiloglou. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
- [89] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin. Large linear classification when data cannot fit in memory. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 833–842, New York, NY, USA, 2010. ACM.
- [90] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 271–278, New York, NY, USA, 2007. ACM.
- [91] D. Zhang, R. Mao, H. Li, and J. Mao. How to count thumb-ups and thumb-downs?: an information retrieval approach to user-rating based ranking of items. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 1223–1224, New York, NY, USA, 2011. ACM.

Index

- Bag-of-words model, 25
- Centroid based classifier, 39
 - Arithmetical Average Centroid, 39
 - Class-Feature-Centroid classifier, 39
 - code book, 40
 - Cumuli Geometric Centroid, 39
 - prototype, 40
- Formal Concept Analysis, 102
 - concept dis-similarity, 113
 - concept lattice, 108
 - concept ring, 107
 - Formal Context, 107
- Impurity measures, 26
 - Classification error, 26
 - Entropy, 26
 - Gini, 26
- Jensen-Shannon divergence, 47
- Kullback-Leibler divergence, 27
- Math markup language, 21
 - ASCIIMath, 21
 - MathML, 21
 - OMDoc, 21
 - OpenMath, 21
- Online Learning, 17
 - Bandit, 17
 - Confidence Weighted Algorithm, 19
 - Mahalanobis Passive Aggressive, 64
 - Passive Aggressive Algorithm, 17
 - Perceptron, 17
 - Ranking Passive Aggressive, 92
 - Second-order Perceptron, 18
- Supervised term weighting, 11
 - Chi squared, 11
 - Relevant Frequency, 11
 - Weighed Frequency and Odds, 11
- Text categorization, 14
 - Class-Feature-Centroid classifier, 16
 - machine learning, 15
 - decision tree, 15
 - K-Nearest-Neighbor, 15
 - Naïve Bayes, 15
 - Support Vector Machine, 15
- Two-view online learning, 75
 - hybrid model, view agreement, 75
- Unsupervised term weighting, 11
 - inverse document frequency, 11
 - term frequency, 11
- Word Cloud, 42
 - mean, 43
 - moment generating function, 42
 - similarity, 43
 - union, 43