

This is to certify that the
dissertation entitled


**BOOSTING AND ONLINE LEARNING FOR
CLASSIFICATION AND RANKING**

presented by

HAMED VALIZADEGAN

has been accepted towards fulfillment
of the requirements for the

Ph.D degree in Computer Science


Major Professor's Signature

08/27/2010

Date

MSU is an Affirmative Action/Equal Opportunity Employer

**LIBRARY
Michigan State
University**

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.
MAY BE RECALLED with earlier due date if requested.

DATE DUE	DATE DUE	DATE DUE

**BOOSTING AND ONLINE LEARNING FOR CLASSIFICATION
AND RANKING**

By

Hamed Valizadegan

A DISSERTATION

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

DOCTOR OF PHILOSOPHY

Computer Science

2010

ABSTRACT

BOOSTING AND ONLINE LEARNING FOR CLASSIFICATION AND RANKING

By

Hamed Valizadegan

This dissertation utilizes boosting and online learning techniques to address several real-world problems in ranking and classification. Boosting is an optimization tool that works in the function space (as opposed to parameter space) and aims to find a model in batch mode. Typically, boosting iteratively constructs weak hypotheses with respect to different distributions over a fixed set of training instances and adds them to a final hypothesis. Online learning is the problem of learning a model when the instances are provided over trials. In each trial, a new sample is presented to the learner, the learner predicts its class label and then receives some feedback (partial or complete). The learner updates its model by utilizing the feedback and then a new trial starts.

We consider several learning problems, including the usage of side information in ranking and classification, learning to rank by optimizing a well-known information retrieval measure called NDCG, and online classification with partial feedback.

Using side information to improve the performance of learning techniques has been one research focus of machine learning community for the last decade. In this dissertation, we utilize the abundance of unlabeled instances to improve the performance of multi-class classification, and exploit the existence of a base ranker to improve the performance of learning to rank, both using the boosting technique.

Direct optimization of information retrieval evaluation measures such as NDCG and MAP has received increasing attention in the recent years. It is a difficult task because these measures evaluate the retrieval performance based on the ranking list of documents induced by the ranking function, and therefore they are non-continuous and non-differentiable. To

overcome this difficulty, we propose to optimize the expected value of NDCG and utilize boosting technique as the optimization tool.

Online classification with partial feedback is recently introduced and has applications in contextual advertisement and recommender systems. We propose a general framework for this problem based on exploration vs. exploitation tradeoff technique and introduce effective approaches to automatically tune the exploration vs. exploitation tradeoff parameter.

© Copyright by
HAMED VALIZADEGAN
2010

*To my loving parents, Simin Rahimi and Reza Valizadegan, for their
unlimited and unconditional encouragement, support, and love.*

ACKNOWLEDGMENTS

During my Ph.D, I have received support from a number of people without whom the completion of this thesis was not possible.

First of all, I would like to express my deepest gratitude to my thesis advisor, Dr. Rong Jin, for his unique supervision and guidance. He motivated me to work on a diverse set of problems in machine learning and provided me with an excellent mathematical and optimization knowledge support. Under his supervision, I have learned different aspects of conducting high-quality research and become capable of publishing papers in prestigious research venues such as NIPS and WWW.

For a number of years, I have also worked closely with Dr. Pang-Ning Tan with whom I published a few papers in data mining. I would like to present my sincere appreciation for his valuable support during those years. I will never forget his kindness and help.

I would also like to thank my committee members, Dr. Anil K. Jain, Dr. Joyce Chai, and Dr. Selin Aviyente for their valuable feedback and discussions during my compressive and thesis exams.

I want also thank the Department of Computer Science and Engineering at Michigan State University that provides me with the financial support in terms of teaching assistant for a number of semesters. I would like to particularly thank Dr. Abdol-hosseini Esfahian, Dr. Eric Torng and Linda Moore for their amazing attitude in helping graduate students in the department.

The contextual advertisement group of Yahoo! kindly provided me with an exceptional work atmosphere during Summer and Fall 2008. I would like to thank everyone in their group, particularly Dr. Jianchang Mao, the head of contextual and display advertisement science and Ruofei Zhang, my direct mentor.

It has been a great pleasure to collaborate with Dr. Hang Li, the research manager of Information Retrieval and Mining Group at Microsoft Research Asia, and Dr. Shijun Wang

from National Institute of Health with whom I co-authored research papers in ranking and online learning, respectively.

Finally, I should thank the members of LINKS and PRIP labs for all the great supports they have provided me with during my Ph.D. Particularly, I would like to thank Wei Tong, Fenhjie Li, Yang Zhou, Pavan Mallapragada, and Matthew Gerber.

TABLE OF CONTENTS

LIST OF TABLES	xi
LIST OF FIGURES	xii
1 Introduction	1
1.1 Classification	2
1.2 Learning to Rank	3
1.2.1 Training set	5
1.2.2 Evaluation	6
1.2.3 Learning	6
1.3 Batch Learning	7
1.3.1 Boosting	8
1.4 Online Learning	11
1.5 Contribution of This Dissertation	13
1.6 Benchmark Data Sets	15
1.6.1 Classification Data Sets	15
1.6.2 Ranking Data Sets	16
2 Semi-Supervised Multi-Class Boosting	18
2.1 Introduction	19
2.2 Related Work	22
2.3 Multi-Class Semi-supervised Learning	23
2.3.1 Problem Definition	23
2.3.2 Assemble Algorithm	23
2.3.3 Design of Objective Function	25
2.3.4 Multi-Class Boosting Algorithm	27
2.4 Experiments	31
2.4.1 Experimental Setup	32
2.4.2 Evaluation of Classification Performance	33
2.4.3 Sensitivity to the Combination Parameter C	36
2.4.4 Sensitivity to Base Classifier	36
3 Optimizing NDCG Measure by Boosting	40
3.1 Introduction	41
3.2 Related Work	43
3.3 Optimizing NDCG Measure	44
3.3.1 Notation	44
3.3.2 AdaRank Algorithm	45

3.3.3	A Probabilistic Framework	46
3.3.4	Objective Function	48
3.3.5	Algorithm	50
3.4	Experiments	54
3.4.1	Experimental setup	55
3.4.2	Results	56
4	Ranking Refinement by Boosting	58
4.1	Introduction	58
4.2	Related Work	61
4.3	Ranking Refinement	62
4.3.1	Problem Definition	62
4.3.2	Encoding Ranking Information	63
4.3.3	Objective Function	64
4.3.4	Boosting Algorithm for Ranking Refinement	69
4.4	Experiments	74
4.4.1	Experimental Setup	74
4.4.2	Results for Relevance Feedback	77
4.4.3	Effect of Base Ranker	78
4.4.4	Effect of Size of Feedback Data	79
4.4.5	Results for Recommender System	79
4.4.6	Time Efficiency of Ranking Refinement	80
5	Online Classification with Bandit Feedback	85
5.1	Introduction	86
5.2	Related Work	87
5.3	A Potential-based Framework for Classification with Partial Feedback	88
5.3.1	Problem Definition	88
5.3.2	Banditron	90
5.3.3	Potential-based Online Classification for Partial Feedback	90
5.3.4	Exponential Gradient for Online Classification with Partial Feedback	95
5.4	Experiments	97
5.4.1	Experimental results	100
6	Robust Online Classification With Bandit Feedback	102
6.1	Introduction	102
6.2	Related Work	105
6.3	Balancing between Exploration and Exploitation	106
6.3.1	Preliminary	106
6.3.2	Finding Optimal γ using $[\hat{y}_t \neq y_t] \leq \tau_t$ and $[\hat{y}_t = y_t] \leq \mu_t$	108
6.3.3	Finding Optimal γ using $[\hat{y}_t \neq y_t] \leq 1$ and $[\hat{y}_t = y_t] \leq \mu_t$	110
6.3.4	Finding Optimal γ using $[\hat{y}_t \neq y_t] \leq \tau_t$ and $[\hat{y}_t = y_t] \leq 1$	111
6.4	Experiments	112
6.4.1	Experimental Settings	112
6.4.2	Experimental results	113

7 Conclusion and Future Work	116
7.1 Summary and Conclusions	116
7.1.1 Boosting	116
7.1.2 Online Learning	118
7.2 Future Work	119
7.2.1 Boosting	119
7.2.2 Online learning	120
APPENDICES	122
A APPENDIX	123
A.1 Proof of Lemma 1, Chapter 2	123
A.2 Proof of Lemma 2, Chapter 2	124
A.3 Proof of Theorem 4, Chapter 2	125
A.4 Proof of Proposition 2, Chapter 3	126
A.5 Proof of Lemma 4, Chapter 3	126
A.6 Proof of Theorem 5, Chapter 3	127
A.7 Proof of Theorem 6, Chapter 3	127
A.8 Proof of Theorem 7, Chapter 3	128
A.9 Proof of Theorem 8, Chapter 4	130
A.10 Proof of Lemma 5, Chapter 4	131
A.11 Proof of Theorem 9, Chapter 4	131
A.12 Proof of Theorem 10, Chapter 4	132
A.13 Proof of Proposition 5, Chapter 5	133
A.14 Proof of Theorem 11, Chapter 5	133
A.15 Proof of Lemma 7, Chapter 5	135
A.16 Proof of Theorem 14, Chapter 6	135
A.17 Proof of Proposition 6, Chapter 6	136
A.18 Proof of Proposition 7, Chapter 6	136
A.19 Proof of Proposition 8, Chapter 6	137
BIBLIOGRAPHY	138

LIST OF TABLES

1.1	Description of the classification data sets used in this dissertation	16
1.2	Description of data sets in Letor 3.0.	17

LIST OF FIGURES

2.1	Performance comparison	35
2.2	Sensitivity to parameter C	37
2.3	Sensitivity to the base ranker	39
3.1	The experimental results in terms of NDCG for Letor 3.0 data sets	57
4.1	Reduction of the objective function L_p using the OHSUMED Data Set	71
4.2	NDCG of relevance feedback for different algorithms	81
4.3	NDCG of MRR with different base rankers for relevance feedback	82
4.4	NDCG of MRR with different numbers of feedback	83
4.5	The ranking result for recommender system	84
4.6	Running time of MRR for different numbers of movies	84
5.1	Performance comparisons of different methods	98
5.2	Performance comparisons of different methods with varied γ	99
6.1	The error rates of Banditron with different choice of γ	104
6.2	The error rates of different methods over trials	114

Chapter 1

Introduction

Learning is the task of constructing a prediction model using training data. A learning task is defined by an objective function that evaluates the performance of each model in the domain. A variety of objective functions for learning are defined for different learning tasks. These learning tasks differ in I) their type of prediction, II) the type of feedback/labeling for training data, and III) the way training data are presented to them.

Based on the type of prediction, the learning algorithms can be classified into three major groups: classification, regression, and learning to rank. A **regression** model aims to map an instance to a numerical value. A **classification** model (classifier) categorizes instances into predefined classes and a **ranking** model (ranker) orders a series of items based on a given request.

Training instances can be presented to the learner in two different ways: batch mode and online mode. In **batch mode**, a set of training instances are provided to the learner and the learner trains a model off-line. The learned model is evaluated based on the prediction made for unseen test instances. We usually assume the training instances are i.i.d samples from an unknown distribution and the objective is to learn a statistical model that is able to make accurate prediction for unseen instances sampled from the same distribution of the training data. In **online mode**, the task of learning and making prediction are performed

at the same time; i.e. the learner applies the current model to each received instance, and then receives the feedback for that instance and consequently updates the model based on the instance and the feedback. In online mode, we do not have to make the i.i.d assumption regarding the received instances and the data generator produces instances arbitrarily [1].

The **feedback** for the training instances can be either **partial or full** in online mode and the **label** for training instances can be either **present or absent** in batch mode. Each of these combination results in different learning tasks. When we discuss batch learning in more details in section 1.3, we cover a brief description of semi-supervised learning, in which part of training instances are unlabeled; we discuss online learning with partial feedback in Section 1.4 where the feedback only indicates if the predicted class is correct.

In the following sections, we focus on classification, learning to rank, batch and online learning to draw the direction of materials in the future chapters of this thesis.

1.1 Classification

Classification is the task of categorizing instances into predefined classes and has found countless number of applications. In the fully supervised mode, the learning algorithm receives a set of labeled instances, each represented by a vector of features and a label that shows its class assignment. The objective of the learning algorithm is to learn a classifier that is able to make accurate prediction for unseen examples, generated by the same distribution for training instances. The ability of a learner in producing models that perform well for unseen instances is called **generalization ability** [2] in the machine learning literature. Many effective algorithms have been proposed for the task of supervised classification, such as Support Vector Machines (SVMs) [3], logistic regression [2], and boosting [4].

Classification is one of the oldest machine learning tasks. Nonetheless, it still finds applications that demands developing new techniques. One of the major challenges we address in this dissertation is to learn a classification model from partial feedbacks. As

an example, consider the problem of contextual advertisement that chooses advertisements to display on a web page for a specific user [5]. The contextual advertisement algorithms are usually based on this assumption that users provide feedback by clicking on relevant advertisements [5]. However, if none of the displayed advertisements are relevant to the user's information needs, they will not be clicked and consequently the algorithm does not know which advertisements are relevant for the user. We refer to this scenario as partial feedback as opposed to the case of full feedback where the correct output (i.e., the relevant advertisement) is provided for each instance. This task demands new online learning algorithms that are able to learn over the trials in the partial feedback setting. In particular, the online algorithms need to explore the exploration vs. exploitation trade-off techniques that are primarily developed for multi-armed bandit problem [6].

The performance of a classification algorithm is usually evaluated by the classification accuracy. For the evaluation of multi-class or multi-label learning, the classification accuracy may not be sufficient, particularly when the number of classes is large or classes are unbalanced. In those cases, the most commonly measures used for classification are precision, recall or a combination of these two, such as F1 measure and ROC curve.

1.2 Learning to Rank

Ranking is the task of ordering a list of offerings for a given request. It receives a set of offerings and a request as input and outputs the list of offerings sorted according to their relevancy to the request. The performance of a ranking algorithm is evaluated based on how well it sorts the offering according to their relevancy to the request. **Learning to rank** is the task of learning a ranking function that can order the offerings for unseen requests. It receives a set of requests, each with a sorted list of offerings as the training set and produces a ranking function to sort offerings for new requests. Learning to rank is a relatively new area of study in machine learning that has received much attention in recent years because

of its important role in a variety of applications including:

- **Document Retrieval:** In document retrieval, the request is a textual query (a set of keywords) and the offerings are documents. Users provide a set of keywords to the system and the ranking system should retrieve the most relevant documents to those keywords.
- **Recommender Systems:** In recommender systems, the request is a user and the offerings are the items to be recommended. For example, in movie recommendation system, a ranking system aims to recommend the most interesting movies to a particular user based on the history of users and movies information.
- **Sentiment Analysis:** In sentiment analysis, the request is a text and the offerings are the attitudes of the author regarding to a particular subject.
- **Computational Biology:** In computational biology, a request is a protein and the offerings are the list of different 3d structures. The objective is to provide a sorted list of 3d structures for a given protein.
- **Online Advertisement Placement:** In online advertisement placement, the request is a user visiting a web page and the offerings are the advertisements. Online advertisement systems should rank the relevancy of different advertisements to that user and display the most relevant advertisement on the web page in order to maximize the number of clicks on the advertisements.

Throughout this thesis, we use the document retrieval terminology (e.g. query for request, document for offering) when talking about ranking although the material are applicable to other domains. Since learning to rank is a relatively new problem, we describe it in more details here. A learning to rank system usually consists of three components that distinguish it from classification and regression.

1.2.1 Training set

The training set for learning to rank consists of a set of queries. For each query, a list of documents and their relevancy to the query are provided. The common practice in learning to rank is to assume the existence of a set of base rankers that can be considered the feature generators for query-document pairs. PageRank [7], vector space model [8], and statistical language models [9] such as BM25 are some example base rankers. These base rankers are basically unsupervised models that measure the relevancy of each document to a query. The value produced by each base ranker is considered a feature for a query-document pair and the learning to rank algorithm aims to combine these feature values to produce a ranking function.

The label information in learning to rank is in form of relevancy judgments that can be of three different types: relevancy scores, pairwise relevancy information (partial ordering) and a complete ordering. A **relevancy score** is a numerical value (e.g. 1,2,..) that shows the level of relevancy of documents to a given query [10]. Relevancy scores are the most widely used relevancy information. The **pairwise relevancy information** is the relative relevancy between two documents that indicates which document among the two is more relevant. The pairwise relevancy can often be derived from the implicit feedbacks from users. For example, in search engines, when a user clicks on one of the ranked documents, it is safe to infer that the clicked document is more relevant than the documents that are ranked before the clicked one. This type of **click-through feedback** provides the relative relevancy for pairs of documents [11]. A less commonly used relevancy information is a **complete relevancy ordering** of documents to a given query [12] in which documents are ordered in the descending relevancy. Notice that the relevancy scores can be converted to a pairwise ordering and complete ordering but the opposite is not true.

1.2.2 Evaluation

The performance of a ranking system is evaluated based on how well it predicts the relevancy of documents to a query. Several evaluation measures are introduced in the literature. Area under the ROC Curve (AUC), Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) are some of the most-widely used measures. AUC is based on the Wilcoxon test, a nonparametric statistical test to measure the distributional difference between two sets of numbers. AUC works only for two levels of relevancy judgments and measures how well a ranking function places the relevant documents on the top of the irrelevant documents. AUC treats documents similarly regardless of their position in the ordered list. However, the top retrieved documents are more important because users only look for the relevant documents at the top of the list (e.g. consider a search engine in which users only look at the first few pages of retrieved links). Based on this observation, MAP [13] and NDCG [14] are constructed to put more weight on the documents at the top of the list. Similar to AUC, MAP only works for binary relevancy judgment. On the other hand, NDCG is a general evaluation measure that can handle ranking problems with multiple levels of relevancy judgements.

1.2.3 Learning

Three types of learning to rank algorithms can be found in the literature: Pointwise, pairwise and listwise approaches. **Pointwise approaches** [15–17] can be applied when the relevancy scores of documents are available. In this case, the relevancy scores are considered as absolute quantities and a classification or regression technique is applied by treating the relevancy scores as class labels or numerical values. The **pairwise approaches** are the only group of techniques that can handle the pairwise relevancy information. They apply a classification or regression technique to learn the ordering information of pairs of documents [18–23]. The third group of algorithms, the listwise approaches, are the most effective learning to rank techniques that have been studied in the last few years. They are

motivated by this observation that most evaluation metrics of information retrieval measure the ranking quality for individual queries, not documents. These approaches consider the ranking list of documents for every query as a training instance [13, 24–29] by optimizing a listwise loss function. We describe these techniques in more details in Chapter 3.

1.3 Batch Learning

In batch learning, a set of training instances are provided that are generated by an unknown distribution. The goal is to train a model off-line that is capable of making accurate prediction for unseen instances. As mentioned before, dependent on the type of training instances and their labels, different learning tasks can be defined. For example, in classification, each instance is a vector of features and the label is the class assignment. And in the listwise approach to learning to rank, each instance consists of a query, the list of its documents, and the relevancy of documents to the query.

Training instances can be either all labeled or partially labeled that results in two different modes of learning: **supervised** and **semi-supervised** learning. All training instances are labeled in supervised learning and plenty of unlabeled instances are provided in case of semi-supervised learning to help the process of learning. The usage of unlabeled instances are based on some assumptions about the data generating process such as manifold and cluster assumption [30–35]. We return to these assumptions in Chapter 2.

In most studies of batch learning, an **objective function** is designed to measure the performance of a given model (function) on instances. Different learning algorithms can be designed by defining different objective function for the same task. For example, in case of classification, the negative log-likelihood function is used in logistic regression, a hinge loss leads to support vector machines, and etc. In case of learning to rank, the pointwise approaches utilize a classification or regression model, i.e. they utilize a classification or regression loss function. Similarly, pairwise approaches are concluded from designing a

classification or regression model on pair of documents and a listwise learning to rank algorithm results from utilizing a loss function in the level of query.

Given an objective function (loss function) $L(F)$ to measure the performance of a given model F , learning translates to the process of finding F that optimizes $L(F)$. A common approach is to restrict the model to a member of a parametric family $F(w)$ (e.g. a linear model). This constraint translates the objective function $L(F)$ into an objective function of parameters w , i.e. $L(w)$, and consequentially the optimal model is found by optimizing the objective function with respect to w . In this case, $L(w)$ is called a function in the **parameter space**. A different approach is to directly optimize L over function F . This approach optimizes the objective function in the **function space** and is called boosting. Boosting is the optimization technique we utilize in this thesis for the batch mode algorithms we cover.

1.3.1 Boosting

Boosting [4, 36] is a popular technique with a greedy nature designed to optimize a given objective function in the space of functions. This is very important because it allows to boost the performance of any base function (weak learner) once the problem is written in the function space. Boosting can be considered as a gradient descent algorithm applied in the function space [37]; in each step i , it learns a new direction f_i and a step size α_i to move as much as possible toward the optimum point, which results in a final solution of $F_n = \sum_{i=1}^n \alpha_i f_i$. Instead of applying a direct optimization approach such as gradient descent, bound optimization strategies [38] may be used; this is because f_i and α_i are dependent on each other and it is difficult to decide the values for f_i and α_i simultaneously. The bound optimization strategy is often applied to decouple the dependency between f_i and α_i . We use this technique in different parts of this thesis.

First introduced by Schapire [4], boosting was initially designed to convert a weak learner that performs just slightly better than random guessing into an accurate classifier. Here, by random guessing, we mean a classifier with less than 50% classification error.

However, as we will show throughout this dissertation, the meaning of random guessing can change from one problem to another. In this view, given a set of labeled training examples $(x_i, y_i), i = 1..n$, a boosting algorithm provides the weak learner with a set of weighted training examples at each round. The weak learner constructs a model by optimizing its loss over the weighted training examples. In the new iteration, the boosting algorithm produces a new set of weighted examples by increasing the weights for the examples that are misclassified in the previous round. The iterations are repeated till the algorithm converges.

One well-known boosting algorithm is AdaBoost [39], developed based on an exponential loss function for classification. Algorithm 1 shows AdaBoost algorithm. At the beginning of this algorithm, the booster chooses a uniform weighting over the examples (Step 3). Given the weights produced by the booster, the weak learner constructs a binary classifier that minimizes the loss ϵ_t at Step 5. The booster then produces a new set of weights for the examples in Step 8 by increasing the weights for the examples misclassified in the previous round of learning (Steps 6 and 7). These steps are repeated for a number of times. We have the following bound for the misclassification error of the final hypothesis generated by AdaBoost algorithms:

$$\epsilon \leq 2^T \prod_{i=1}^T \sqrt{\epsilon_t(1 - \epsilon_t)} \quad (1.1)$$

where ϵ_t is the classification error for the hypothesis generated in round t . The above result shows that, under the assumption of weak classifier, the classification error is guaranteed to be reduced as the iteration proceeds.

Using the Minimax theorem, Freund et al. [39] showed that there is a mixed strategy over the space of hypotheses H that produces zero classification error over the training set if (H, X) is γ -learnable. The progress of a boosting algorithm is measured by how much the classification error (or a given loss) decreases at each iteration (or over time) and

For $\gamma > 0$, a learning algorithm is γ -learnable if for any distribution Q over training examples X , the algorithm can return $h \in H$ with at most $\frac{1}{2} - \gamma$ classification error

Algorithm 1 AdaBoost Algorithm

1: Input

1. A weak learner, and a set of training examples
2. A set of training examples $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X$ and $y_i \in \{-1, 1\}$.

2: Initialize $F(x_i) = 0, i = 1, \dots, m$

3: Initialize $D_1(i) = 1/m, i = 1, \dots, m$

4: repeat

5: Find the classifier $f_t : X \rightarrow \{-1, 1\}$ that minimizes $\epsilon_t = \sum_{i=1}^m D_t(i) I(y_i \neq f_t(x_i))$

6: Compute $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$

7: Compute $F(x_i) = F(x_i) + \alpha f_t(x_i), i = 1, \dots, m$

8: Compute the new weighting $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i f_t(x_i))}{Z_t}$ where Z_t is the normalization factor.

9: **until** reach the maximum number of iterations

defined in the following form:

$$M(P_t, Q_0) \leq \Pi_{t=1}^T \delta(M(h_t, Q_t)) \quad (1.2)$$

where δ is an increasing function of the loss, $M(P_t, Q_0)$ is the suffered loss when the majority vote P_t is used over H and Q_0 is the uniform distribution over X (i.e. $M(P_t, Q_0)$ is the computed loss of weighted majority vote over the original samples), and $M(h_t, Q_t)$ is the computed loss at round t (i.e. the loss suffered when a single hypothesis h_t is applied over the weighted samples set Q_t).

Beside classification and regression, boosting has been applied to a wide range of applications including:

- **Semi-Supervised Learning:** Boosting can be utilized to adapt a supervised learner to the problem of semi-supervised learning. For example, [40] used binary classifier as the weak learner and boosted it for the task of semi-supervised classification and [41] exploited a binary supervised learner as the weak learner and boosted it for semi-supervised clustering.

- **Learning to Rank:** Boosting is used to learn a ranking function to order the relevancy of documents for a query. RankBoost [19] and AdaRank [42] are example applications of boosting to ranking. RankBoost uses pairwise binary classifier and boost it for ranking and AdaRank adapts AdaBoost to optimize information retrieval evaluation measures such as Normalized Discounted Cumulative Discount (NDCG) and Mean Average Precision (MAP).

1.4 Online Learning

Online learning is the task of learning when the examples are provided sequentially (over the trials). In each trial, the learning algorithm receives a new example, classifies it and then acquires some sort of feedback. Using this feedback, the online learning algorithm updates the model in order to better classify the future examples. The feedback provided to the online algorithm can be either full or partial. In the **full feedback setting**, after classifying an instance, the algorithm receives its true class label. One well-known example of such online learning algorithm is the well-known Perceptron algorithm [43]. In the **partial feedback** or "**Bandit**" **setting**, the true label is not revealed and the feedback is limited to whether or not the algorithm classified the instance correctly. Since the difference between full and partial feedback in the above discussion only makes sense for the case of multi-class classification, the online classification with partial feedback is called multi-class bandit learning [5]. The objective of the learner is to generate a sequence of hypotheses that guarantees a small cumulative loss in the long run when compared to the best hypothesis in the space of hypothesis; i.e.

$$\frac{1}{T} \sum_{t=1}^T M(P_t, Q_t) \leq \frac{1}{T} \min_P \sum_{t=1}^T M(P, Q_t) + \delta(T) \quad (1.3)$$

where δ is a decreasing function of T and should approaches zero when T approaches infinity.

The bandit feedback has several real-world applications such as online advertisement [5] and recommender systems [5], as described in the following

- **Online Advertisement:** In online advertisement, we often assume that a sponsored ad is likely to be relevant to the user's query if it is clicked by the user, and irrelevant otherwise. In the case when the sponsored ad does not receive a click, the online advertisement algorithm is unable to locate the advertisements that are relevant to the given query, leading to the partial user feedback.
- **Recommender Systems:** A recommender system recommends some items (e.g. movies) to the user. The assumption is that if one of the recommended movies are selected by user, that movie was a correct recommendation. However, if none of the recommended movies are chosen by the user, the recommender system is not able to discover the right set of movies for that user.

While the problem of online classification with full feedback is well-studied, online classification with bandit feedback has received attention only recently [5]. Kakade et al. [5] introduced Banditron as an extension to Perceptron [43] to handle the partial feedback setting. Online learning with bandit feedback can be regarded as the problem of multi-armed bandit [44] when some side information (e.g. the feature vector of instances) is available. Multi-armed bandit is the generalized version of one-armed bandit game (a traditional slot machine) in which several levers are provided and the player aims to choose a lever that maximizes the rewards in the long run. At each stage, the player only knows the reward for the lever he chooses; the rewards for the remaining levers are unknown to the player. In a more abstract level, multi-armed bandit problem refers to the problem of choosing an action from a list of actions to maximize rewards given that the feedback is (bandit) partial. The algorithms developed for this problem usually utilize the exploitation vs. exploitation

tradeoff strategy to handle the challenge arising from partial feedback [45–47].

1.5 Contribution of This Dissertation

We address several important ranking and classification problems in this dissertation. Utilizing side information in ranking and multi-class classification, direct optimization of information retrieval measures such as NDCG, and online learning in the bandit setting are the subjects we cover, as summarized here:

- **Semi-supervised Classification:** The focus of semi-supervised classification is on constructing better models by utilizing unlabeled instances when the number of labeled instances is small. Several semi-supervised classification algorithms are developed based on manifold [32–35] and cluster [30, 48, 49] assumptions. Most of these techniques work for binary problems and converting techniques such as one-versus-one and one-versus-the-rest are applied to use them for multi-class problems [50]. This converting procedures has several well-known problems including imbalanced classification and different output scales of different binary classifiers. We utilize both manifold and cluster assumptions in Chapter 2 and design an objective function that directly addresses multi-class semi-supervised problem. We solve this objective function in the function space using boosting technique. Our empirical study shows the superior performance of this boosting algorithm compared to the existing boosting algorithms for multi-class problems.
- **Ranking by optimizing NDCG:** The objective in this problem is to learn a ranking function by maximizing Normalized Discounted Cumulative Gain (NDCG), the most frequently used information retrieval evaluation measure for ranking problems with multi level relevance judgement [10]. This is a difficult problem because NDCG is a non-differentiable and non-continuous loss function. In order to overcome this difficulty, we introduce the expected value of NDCG and solve it in the function space

using the boosting technique. The detailed discussion of this boosting algorithm is provided in Chapter 3.

- **Ranking Refinement:** In some real world applications, there are two complementary sources of information for ranking, ranking information given by the existing ranking function (i.e., the base ranker) and that obtained from users feedback. One example of such applications is relevance feedback, where the two sources of information are the relevance scores obtained from a ranking function like BM25 [51] and the relevance judgments obtained by the users. The key challenge in combining the two sources of information arises from the fact that the ranking information presented by the base ranker tends to be imperfect and the ranking information obtained from users' feedbacks tends to be noisy. We encode these sources of relevancy information in form of pairwise relevancy and design an objective function to combine them. We also design a boosting algorithm to solve the resulting objective function. The detailed discussion is provided in Chapter 4 where we perform extensive experiments to show the superiority of our proposed framework to several baselines.
- **Online Multi-class Learning with Partial Feedback:** Unlike online learning with complete feedback that has been extensively studied [52], the problem of online multi-class learning with bandit feedback was introduced very recently [5]. Banditron, the first introduced algorithm for multi-class learning with bandit feedback, is a direct generalization of Perceptron to the case of partial feedback that uses exploration vs. exploitation tradeoff strategy to handle partial feedback [5]. Using potential function and exploration vs. exploitation tradeoff technique, we develop a general framework in Chapter 5, of which Banditron is a special case. The major problem with Banditron is that its performance could be sensitive to the parameter that trades off between exploration and exploitation [53]. We develop an effective approach in Chapter 6 to reduce this dependency.

1.6 Benchmark Data Sets

Throughout this dissertation, we use two sets of data to study the performance of the proposed methods, one set for multi-class classification and one set for learning to rank, as described in the following subsections. We use 5 folds cross validation to run all the experiments except for online learning.

1.6.1 Classification Data Sets

Multiple benchmark data sets from UCI data repository [54] and LIBSVM web page [55] are used in our study. Here is the list and a brief description of these data sets:

- **MNIST.** MNIST is comprised of grey scale images of size 28×28 for hand written digits. It contains 60000 training samples, each represented by 780 features.
- **Protein.** Protein has 17766 samples, represented by 357 features and three classes.
- **Letter.** Letter contains 15000 instances of 26 characters, each represented by 16 features.
- **optdigits.** This data set consist of normalized bitmaps for handwritten digits from 30 people. It contains 3823 instances, each represented by 64 features.
- **pendigits.** This is another collection of images for handwritten digits. It contains 7495 samples, each represented by 16 features.
- **Nursery.** Originally developed to rank applications for nursery school, it has 12960 records, each represented by 8 features belonging to one of 4 classes (we removed one class that only had two samples).
- **Isolet.** Isolet contains 7797 spoken alphabet that belong to 26 classes, with each letter forming its own class. Every spoken alphabet is represented by 617 attributes.

Notice that for some of these data sets, there were two separate sets, one for training and one for testing. We only used the training set in our experiments. The information related to these data sets are summarized in Table 1.1.

Table 1.1: Description of the classification data sets used in this dissertation

	Instances	Features	Classes
Isolet	7797	617	26
MNIST	60000	784	10
Protein	17766	357	3
Optdigits	3823	64	26
Nursery	12960	8	3
Letter	15000	16	26
Pendigits	7495	16	26

1.6.2 Ranking Data Sets

We use data sets from **information retrieval** and **recommender systems** to study the performance of ranking algorithm in our studies. For **information retrieval**, we use version 3.0 of LETOR package provided by Microsoft Research Asia [56]. LETOR Package includes several benchmark data sets for ranking, along with the state-of-the-art algorithms for learning to rank and tools for evaluation. There are seven data sets provided in the LETOR package: OHSUMED, Top Distillation 2003 (TD2003), Top Distillation 2004 (TD2004), Homepage Finding 2003 (HP2003), Homepage Finding 2004 (HP2004), Named Page Finding 2003 (NP2003) and Named Page Finding 2004 (NP2004). There are 106 queries in the OSHUMED data sets, with each query equipped with around 1000 manually judged documents. The relevancy of each document in OHSUMED data set is scored in three levels: 0 (irrelevant), 1 (possibly) or 2 (definitely). The total number of query-document relevancy judgments provided in OHSUMED data set is 16140 and there are 45 features used to represent each document-query pair . For TD2003, TD2004, HP2003, HP2004, NP2003, and NP2004 there are 50, 75, 150, 75 150 and 75 queries, respectively, with about 1000 retrieved documents that are manually judged for each query. This amounts to a total number of 49058, 74170, 147606, 148657 and 73834 query-document pairs for TD2003, TD2004, HP2003, HP2004 and NP2003 respectively. For these data

Unlike the classical supervised learning, in learning to rank, the representation of documents depends on the given query. Hence, features are extracted for each document-query pair, not just for individual documents

Table 1.2: Description of data sets in Letor 3.0.

	Query document pair	Queries	Relevancy level	Features
OHSUMED	16140	106	3	45
TD2003	49058	50	binary	63
TD2004	74170	75	binary	63
HP2003	147606	150	binary	63
HP2004	74409	75	binary	63
NP2003	148657	150	binary	63
NP2004	73834	75	binary	63

sets, there are 63 features extracted for every query-document pair. A binary relevancy judgment is provided for every query-document pair. This information is summarized in Table 1.2.

For every data sets in LETOR, five partitions are provided to conduct the five-fold cross validation, and each partition is further divided into the training set, testing set, and validation set. The retrieval results for a number of state-of-the-art learning to rank algorithms are also provided in the LETOR package. We will describe these algorithms in details in Chapter 3.

In order to evaluate the performance of the proposed ranking algorithms for **Recommender System**, we use the MovieLens dataset, available at [57], which is one of the most popular data sets for the evaluation of information filtering. It contains 100,000 ratings ranging from 1 to 5, with 1 as the best rating and 5 as the worst rating for 1682 movies given by 943 users. Each movie is represented by 51 binary features: 19 features are derived from the genres of movies and the rest 32 features are derived from the keywords that are used to describe the content of movies. To extract the content features, we downloaded the keywords of each movie from the online movie database IMBD and selected the keywords that are mostly used by the 1682 movies.

Chapter 2

Semi-Supervised Multi-Class Boosting

Most semi-supervised learning algorithms are designed for binary classification. They are extended to multi-class classification by approaches such as one-against-the-rest. The main shortcoming of these approaches is that they are unable to exploit the fact that each example is only assigned to one class in the case of multi-class learning. Additional problems with extending semi-supervised binary classifiers to multi-class classification include imbalanced classification and different output scales of different binary classifiers. Given that there are well-known multi-class classification techniques such as decision tree and multi-layer perceptron, the research question is whether it is possible to use these techniques as weak learner and boost their performance for the task of semi-supervised learning. The main challenge in designing such boosting algorithms is that the definition of the loss for unlabeled examples is not clear. One approach is to generalize the notion of margin for labeled instances to unlabeled instances. This approach computes the margin for unlabeled examples by considering their assigned labels at the current iteration of the algorithm. However, since the labels computed in the early iterations is likely to be inaccurate, this strategy produces undesirable results.

Unlike the existing boosting algorithms for semi-supervised learning which are only based on the classification confidence (margin) of the examples (i.e. cluster assumption),

we utilize both the classification confidence and the similarity among examples (i.e. the manifold assumptions) to design a loss function for multi-class semi-supervised learning. We further develop a boosting algorithm for efficient computation. Empirical study with the multiple benchmark datasets shows that the proposed MCSSB algorithm performs better than the state-of-the-art boosting algorithms for semi-supervised learning.

2.1 Introduction

Semi-supervised classification combines the hidden structural information in the unlabeled examples with the explicit classification information of labeled examples to improve the classification performance. Many semi-supervised learning algorithms have been studied in the literature. Examples are density based methods [30, 31], graph-based algorithms [32–35], and boosting techniques [40, 48, 49]. Most of these methods are based on either manifold assumption [32–35] or cluster assumption [30, 48, 49]. Under the manifold assumption, the data is assumed to reside on a low dimensional manifold within the original high dimensional space and the class assignment of unlabeled examples can be derived from a classification function that lives in this low dimensional manifold. Under the cluster assumption, the examples of the same class tends to be closer to each other than those of different classes. As a result of this assumption, the decision boundary is expected to pass through the low density regions. Thus, a given semi-supervised learning is usually specified by a combination of two terms, with one term related to the classification error on the training examples and the other term related to how well the model satisfies the assumption (either manifold or cluster assumption).

While most of semi-supervised classification approaches were originally designed for two class problems, many real-world applications, such as speech recognition and object recognition, require multi-class categorization. To adopt a binary (semi-supervised) learning algorithm to problems with more than two classes, a common practice is to divide a

multi-class learning problem into a number of independent binary classification problems using techniques such as one-versus-the-rest, one-versus-one, and error-correcting output coding [58]. The main shortcoming with these approaches is that the resulting binary classification problems are *independent*. As a result, these approaches are unable to exploit the fact that each example can only be assigned to one class. This issue was already pointed out in the study of multi-class boosting [59]. In addition, since every binary classifier is trained independently, their outputs may be on different scales, making it difficult to identify the most likely class assignment based on the classification scores [60]. Though calibration techniques [61] can be used to alleviate this problem in supervised classification, it is rarely used in semi-supervised learning due to the small number of labeled training examples. Moreover, techniques like one-versus-the-rest, where the examples of one class are considered against the examples of all the other classes, could lead to the imbalanced classification problem. Although a number of techniques have been proposed for supervised learning in multi-class problems [59, 62, 63], none of them addressed semi-supervised multi-class learning problems, which is the focus of this chapter.

Given that the supervised multi-class classification is a well-studied subject, an important research question is whether it is possible to develop a general semi-supervised framework that is able to improve the accuracy of a given supervised multi-class learning algorithm by effectively exploring the abundance of unlabeled data. The immediate answer to this question is boosting technique. The objective of semi-supervised classification is to learn a hypothesis that makes minimum number of misclassification on the labeled examples and utilizes the unlabeled data for a better generalization. Given a loss function for the labeled and unlabeled examples, a boosting algorithms can be defined by reweighting each instance based on the current value of the loss.

One straightforward approach to define the loss for unlabeled examples is to consider the classification confidence as the loss for unlabeled instances. The difficulty comes from the fact that the classification confidence related to the unlabeled examples are unknown.

One approach to address this problem is to use the class labels predicted by the current model as the pseudo-labels for the unlabeled examples and utilize them to obtain the classification confidence (or margin). Assemble [48], as described in Section 2.3.2, is constructed based on the idea of pseudo-labels. The problem with utilizing pseudo-labels to compute the loss for unlabeled examples is that the pseudo-labels assigned in the early steps of the algorithm is not precise and can lead to undesirable result of the boosting algorithm. Particularly, this approach does not directly utilize the underlying properties of data described as a manifold or cluster assumption. Moreover, since all the existing semi-supervised boosting algorithms are designed for binary classification, they will still suffer from the aforementioned problems when applied to multi-class problems.

To avoid the above problems, we design a boosting algorithm in this chapter by considering a multi-class loss function that utilizes both the manifold and cluster assumption; i.e. it consists of two terms, one related to the consistency of the predicted labels and similarity between the examples, and one related to the consistency between the predicted labels and the true labels of labeled examples. To minimize this loss function, we develop a semi-supervised boosting framework, termed *Multi-Class Semi-Supervised Boosting (MC-SSB)*, that is designed for multi-class semi-supervised learning problems. By directly solving a multi-class problem, we avoid the problems that arise when converting a multi-class classification problem into a number of binary ones. Moreover, unlike the existing semi-supervised boosting methods that only assign pseudo-labels to the unlabeled examples with high classification confidence, the proposed framework decides the pseudo labels for unlabeled examples based on both the classification confidence and the similarities among examples. It therefore effectively explores both the manifold assumption and the clustering assumption for semi-supervised learning. Empirical study with UCI datasets shows the proposed algorithm performs better than the state-of-the-art algorithms for semi-supervised learning.

2.2 Related Work

Most semi-supervised learning algorithms can be classified into three categories: density based methods [30, 31], graph-based algorithms [32–35], and boosting techniques [40, 48, 49]. As mentioned in Section 2.1, these methods are based on either cluster or manifold assumption, dependent on how they utilize the unlabeled examples. Density-based methods are usually based on finding a decision boundary that passes through sparse regions and have the maximum margin to both labeled and unlabeled examples [30, 31, 48, 49]. Cluster-based learners utilize a similarity measure between examples and construct a graph to propagate the labeling information to the unlabeled instances [32–35].

Semi-supervised learning algorithms can be also categorized into inductive and transductive learner based on their functionality. A semi-supervised learner is called transductive if it does not produce a classifier and cannot operate on the unseen examples. Otherwise, it is called inductive. The algorithm we developed in this chapter works in the inductive mode.

Semi-supervised SVMs (S^3VM s) or Transductive SVMs (TSVMs) are the semi-supervised extensions to Support Vector Machines (SVM). They are essentially density-based methods and assume that decision boundaries should lie in the sparse regions. Unlike their name, TSVMs can work in inductive mode. Although finding an exact S^3VM is NP-complete [64], there are many approximate solutions for it [30, 31, 65–67]. Except for [67], these methods are designed for binary semi-supervised learning. The main drawback with [67] is its high computational cost due to the semi-definite programming formulation.

Graph-based methods are usually transductive learner that aims to predict the class labels that are smooth on the graph of unlabeled examples. These algorithms differ in how they define the smoothness of class labels over a graph. Example graph-based semi-supervised learning approaches include Mincut [32], Harmonic function [33], local and global consistency [34], and manifold regularization [35]. Similar to density based meth-

ods, most graph-based methods are mainly designed for binary classification.

Semi-supervised boosting methods such as SSMBBoost [68] and Assemble [48] are direct extensions of Adaboost [39]. In [49], a local smoothness regularizer is introduced to improve the reliability of semi-supervised boosting. Unlike the existing approaches for semi-supervised boosting that solve 2-class problems, we focused on semi-supervised boosting for multi-class classification.

2.3 Multi-Class Semi-supervised Learning

2.3.1 Problem Definition

Let $\mathcal{D} = (x_1, \dots, x_N)$ denote the collection of N examples. Assume that the first N_l examples are labeled by y_1, \dots, y_{N_l} . Each $y_i = (y_i^1, \dots, y_i^m) \in \{0, +1\}^m$ is a binary vector that indicates the assignment of x_i to m different classes, where. $y_i^k = +1$ when x_i is assigned to the k th class, and $y_i^k = 0$, otherwise. Since we are dealing with a multi-class problem, we have $\sum_{k=1}^m y_i^k = 1$, i.e., each example x_i is assigned to one and only one class. We denote by $\hat{y}_i = (\hat{y}_i^1, \dots, \hat{y}_i^m) \in \mathbb{R}^m$ the predicted class labels (or confidence) for example x_i , and by $\hat{Y} = (\hat{y}_1^\top, \dots, \hat{y}_N^\top)^\top$ the predicted class labels for all the examples. Let $S = [S_{i,j}]_{N \times N}$ be the similarity matrix where $S_{i,j} = S_{j,i} \geq 0$ is the similarity between x_i and x_j . For the convenience of discussion, we set $S_{i,i} = 0$ for any $x_i \in \mathcal{D}$, a convention that is commonly used by many graph-based approaches. Our goal is to compute \hat{y}_i for the unlabeled examples with the assistance of similarity matrix S and $Y = (y_1^\top, \dots, y_{N_l}^\top)^\top$.

2.3.2 Assemble Algorithm

Assemble [48], a boosting algorithm for semi-supervised classification as depicted in Algorithm 2, is constructed based on the idea of pseudo-labels. At each boosting iteration,

x^\top is the transpose of matrix(vector) x .

Algorithm 2 Assemble: Adaptive Semi-Supervised Ensemble Algorithm

1: **Input:**

- $D = (x_1, \dots, x_N)$: The set of examples; the first N_l examples are labeled.
- s : The number of sampled examples

2: Initialize $F(i) = 0, i = 1, \dots, |D|$

3: Initialize $w_1(i) = 1/N_l, i = 1, \dots, N_l$ and $w_1(i) = 0, i = N_l + 1, \dots, |D|$

4: **repeat**

5: Set $y_i = F(x_i), i = N_l + 1, \dots, |D|$

6: Find a multi-class classifier h_t that minimizes $\epsilon_t = \sum_{i=1}^{|D|} w_t(i) I(y_i \neq f_t(x_i))$

7: Compute $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$

8: Compute $F(x_i) = F(x_i) + \alpha f_t(x_i), i = 1, \dots, |D|$

9: Compute the new weighting $w_{t+1}(i) = \frac{w_t(i) \exp(\alpha_t I(y_i \neq f_t(x_i)))}{Z_t}$ where Z_t is the normalization factor and $I(x)$ outputs 1 if x is true, and 0 otherwise.

10: **until** reach the maximum number of iterations

the boosting algorithm creates a new classifier and redistributes the weights by emphasizing more on the less-confident instances.

Beside Assemble, several other boosting algorithms have been proposed for semi-supervised learning based on the idea of using pseudo-labels [49, 68]. They essentially operate like self-training where the class labels of unlabeled examples are updated iteratively: a classifier trained by a small number of labeled examples is initially used to predict the pseudo-labels for unlabeled examples; a new classifier is then trained by both labeled and pseudo-labeled examples; the processes of training classifiers and predicting pseudo-labels are altered iteratively till stopping criterion is reached. The main drawback with this approach is that it relies solely on the pseudo-labels predicted by the classifiers learned so far when generating new classifiers. Given the possibility that pseudo-labels predicted in the first few steps of boosting could be inaccurate, the resulting new classifiers may also be unreliable. This problem was addressed in [49] by the introduction of a local smoothness regularizer. However, these approaches do not utilize the underlying properties of data described as a manifold or cluster assumption. In what follows, we design a boosting algorithm for the problem of multi-class semi-supervised classification based on manifold and cluster assumption.

2.3.3 Design of Objective Function

The goal of semi-supervised learning is to combine labeled and unlabeled examples to improve the classification performance. Therefore, we design an objective function that consists of two terms: (a) F_u that measures the inconsistency between the predicted class labels \hat{Y} of unlabeled examples and the similarity matrix S , and (b) F_l that measures the inconsistency between the predicted class labels \hat{Y} and true labels Y . Below we discuss these two terms in detail.

Given two examples x_i and x_j , we first define the similarity $Z_{i,j}^u$ based on their predicted confidence score \hat{y}_i and \hat{y}_j :

$$Z_{i,j}^u = \sum_{k=1}^m \frac{\exp(\hat{y}_i^k)}{\sum_{k'=1}^m \exp(\hat{y}_i^{k'})} \frac{\exp(\hat{y}_j^k)}{\sum_{k'=1}^m \exp(\hat{y}_j^{k'})} = \sum_{k=1}^m b_i^k b_j^k = b_i^\top b_j \quad (2.1)$$

where $b_i^k = \exp(\hat{y}_i^k) / (\sum_{k'=1}^m \exp(\hat{y}_i^{k'}))$ and $b_i = (b_i^1, \dots, b_i^m)$. Note that b_i^k can be interpreted as the probability of assigning x_i to class k , and $Z_{i,j}^u$, the cosine similarity between b_i and b_j , can be interpreted as the probability of assigning x_i and x_j to the same class. We emphasize it is important to use b_i^k , instead of $\exp(\hat{y}_i^k)$, for computing $Z_{i,j}^u$ because the normalization in b_i^k allows us to enforce the requirement that each example is assigned to a single class, a key feature of multi-class learning.

Let $Z^u = [Z_{i,j}^u]$ be the similarity matrix based on the predicted labels. To measure the inconsistency between this similarity and the similarity matrix S , we define F_u as the distance between the matrices Z^u and S using the Bregman matrix divergence [69], i.e.,

$$F_u = \varphi(Z^u) - \varphi(S) - \text{tr}((Z^u - S)^\top \nabla \varphi(S)), \quad (2.2)$$

where $\varphi : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}$ is a convex matrix function. By choosing $\varphi(X) =$

$\sum_{i,j=1}^N X_{i,j}(\log X_{i,j} - 1)$ [69], F_u is written as

$$F_u = \sum_{i,j=1}^N \left(S_{i,j} \log \frac{S_{i,j}}{Z_{i,j}^u} + Z_{i,j}^u - S_{i,j} \right) \quad (2.3)$$

By assuming that $\sum_{i,j=1}^N Z_{i,j}^u \approx \sum_{k=1}^m N_k^2$ and $\log x \approx x - 1$, where N_k is the number of examples assigned to class k , we simplify the above expression as $F_u \approx \sum_{i,j=1}^N S_{i,j}^2 / Z_{i,j}^u$. Since $S_{i,j}^2$ could be viewed as a general similarity measurement, we replace $S_{i,j}^2$ with $S_{i,j}$ and simplify F_u as

$$F_u \approx \sum_{i,j=1}^N \frac{S_{i,j}}{Z_{i,j}^u} = \sum_{i,j=1}^N \frac{S_{i,j}}{\sum_{k=1}^m b_i^k b_j^k} \quad (2.4)$$

Remark I We did not use $\varphi(X) = \sum_{i,j=1}^N X_{i,j}^2$ [69], which will result in $F_u = \sum_{i,j=1}^N (Z_{i,j}^u - S_{i,j})^2$. This is because the value of $Z_{i,j}^u$ and $S_{i,j}$ may be on different scales which makes it inappropriate to measure the difference between two matrices directly by subtracting their corresponding elements.

Similarly, we define the similarity between a labeled example x_i and an unlabeled example x_j based on their class assignments as follows

$$Z_{i,j}^l = \sum_{k=1}^m y_i^k b_j^k, \quad (2.5)$$

and the label inconsistency measure F_l between the labeled and unlabeled examples as follows:

$$F_l = \sum_{i=1}^{N_l} \sum_{j=1}^N \frac{S_{i,j}}{Z_{i,j}^l} = \sum_{i=1}^{N_l} \sum_{j=1}^N \frac{S_{i,j}}{\sum_{k=1}^m y_i^k b_j^k} \quad (2.6)$$

We can only consider the sub-matrices related to unlabeled examples when defining F_u .

Finally, we linearly combine F_l and F_u to form the objective function:

$$F = F_u + CF_l \quad (2.7)$$

where C weights the importance of F_l . It is set to 10,000 in our experiments to emphasize F_l . Given the objective function F in (2.7), our goal is to find solution \hat{Y} that minimizes F .

2.3.4 Multi-Class Boosting Algorithm

In this section, we present a boosting algorithm to solve the optimization problem in (2.7). Following the general boosting strategy, we incrementally add weak learners to obtain a better classification model. We denote by H_i^k the solution that is obtained for \hat{y}_i^k so far, and by $h_i^k \in \{0, 1\}$ the prediction made by the incremental weak classifier that needs to be learned. Then, our goal is to find $h_i^k, i = 1, \dots, N, k = 1, \dots, m$ and a combination weight α such that the new solution $\tilde{H}_i^k = H_i^k + \alpha h_i^k$ significantly reduces the objective function F in Equation 2.7. For the convenience of discussion, we use symbol \sim to denote the quantities (e.g., \tilde{F}) associated with the new solution \tilde{H} .

The key challenge in optimizing F with respect to h_i^k and α is that these two quantities are coupled with each other and therefore the solution of one variable depends on the solution of the other. Our strategy to solve the optimization problem is to first upper bound F with a simple convex function in which the optimal solution for h_i^k can be obtained without knowing the solution to α . Given the solution to h_i^k , we compute the optimal solution for α . Below we give details for these two steps.

First, the following lemma allows us to decouple the interaction between α and h_i^k within $Z_{i,j}^u$ and $Z_{i,j}^l$

The algorithm is quite stable with different values of C bigger than 1000 according to our experiment.

Lemma 1.

$$\frac{1}{\tilde{Z}_{i,j}^u} \leq \frac{1 + e^{6\alpha} + e^{-6\alpha}}{3Z_{i,j}^u} + \frac{e^{6\alpha} - 1}{3Z_{i,j}^u} \left(\sum_{k=1}^m (b_i^k - \tau_{i,j}^k) h_i^k \right) \quad (2.8)$$

$$\frac{1}{\tilde{Z}_{i,j}^l} \leq \frac{1 + e^{6\alpha} + e^{-6\alpha}}{3Z_{i,j}^l} + \frac{e^{6\alpha} - 1}{6} \sum_{k=1}^m h_j^k \phi_{i,j}^k \quad (2.9)$$

where

$$\tau_{i,j}^k = \frac{b_i^k b_j^k}{\sum_{k'=1}^m b_i^{k'} b_j^{k'}}, \quad \phi_{i,j}^k = \sum_{k'=1}^m y_i^{k'} \frac{b_j^k}{b_j^{k'}} - \frac{y_i^k}{b_j^k} \quad (2.10)$$

The proof of Lemma 1 can be found in Appendix A.1. Using Lemma 1, we derive an upper bound for \tilde{F} in the following theorem.

Theorem 1.

$$\tilde{F} \leq F \frac{1 + \exp(6\alpha) + \exp(-6\alpha)}{3} + \frac{\exp(6\alpha) - 1}{3} \sum_{i=1}^N \sum_{k=1}^m h_i^k (\alpha_i^k + C\beta_i^k) \quad (2.11)$$

where α_i^k and β_i^k are defined as follows:

$$\alpha_i^k = \sum_{j=1}^N \frac{S_{i,j} (b_i^k - \tau_{i,j}^k)}{Z_{i,j}^u}, \quad \beta_i^k = \frac{1}{2} \sum_{j=1}^{N_l} S_{i,j} \phi_{j,i}^k \quad (2.12)$$

Theorem 1 can be directly verified by replacing $1/\tilde{Z}_{i,j}^u$ and $1/\tilde{Z}_{i,j}^l$ in (2.7) with (2.8) and (2.9). Note that the bound in Theorem 1 is tight because by setting $\alpha = 0$, we have $\tilde{H} = H$ and the inequality in Equation 2.11 is reduced to an equality. The key feature of the bound in Equation 2.11 is that the optimal solution for h_i^k can be obtained without knowing the solution for α . This is summarized by the following theorem.

Theorem 2. *The optimal solution for h_i^k that minimizes the upper bound of \tilde{F} in Equa-*

tion 2.11 is

$$h_i^k = \begin{cases} 1 & k = \arg \min_{k'} (\alpha_i^{k'} + C\beta_i^{k'}) \\ 0 & \text{otherwise} \end{cases} \quad (2.13)$$

It is straightforward to verify the result in Theorem 2.

We then proceed to find solution for α given the solution for h_i^k . The following lemma provides a tighter bound for solving α in F .

Lemma 2.

$$\tilde{F} - F \leq (e^{2\alpha} - 1)(A_u + CA_l) + (e^{-2\alpha} - 1)(B_u + CB_l) \quad (2.14)$$

where

$$A_u = \sum_{i,j=1}^N \frac{S_{i,j}}{Z_{i,j}^u} \sum_{k=1}^m h_i^k b_i^k \quad (2.15)$$

$$A_l = \frac{1}{2} \sum_{i=1}^{N_l} \sum_{j=1}^N S_{i,j} \sum_{k,k'=1}^m \frac{y_i^k}{b_j^k} b_j^{k'} h_j^{k'} \quad (2.16)$$

$$B_u = \sum_{i,j=1}^N \frac{S_{i,j}}{Z_{i,j}^u} \sum_{k=1}^m h_i^k \tau_{i,j}^k \quad (2.17)$$

$$B_l = \frac{1}{2} \sum_{i=1}^{N_l} \sum_{j=1}^N S_{i,j} \sum_{k=1}^m y_i^k \frac{h_j^k}{b_j^k} \quad (2.18)$$

The proof of Lemma 2 can be found in Appendix B. Using Lemma 2, Theorem 3 gives the optimal solution for α .

Theorem 3. *The optimal α that minimizes the upper bound of \tilde{F} in Equation 2.14 is*

$$\alpha = \frac{1}{4} \log \left(\frac{B_u + CB_l}{A_u + CA_l} \right) \quad (2.19)$$

Note that this tighter bound can not be used to derive h_i^k

Remark II: Notice that in order to have this boosting algorithm continue working, the weak learner needs to produce models better than random guessing in the following sense.

Writing α in the form

$$\alpha = \frac{1}{4} \log \left(\frac{1 - \epsilon}{\epsilon} \right) \quad (2.20)$$

where

$$\epsilon = \frac{A_u + CA_l}{A_u + B_u + C(A_l + B_u)}$$

can be interpreted as some kind of classification error. Hence, better than random guessing implies $\epsilon \leq 0.5$ in this case.

Algorithm 3 summarizes the proposed boosting algorithm for multi-class semi-supervised learning. In each iteration, MCSSB produces a weighting on the set of training examples based on the evaluation of F . Notice that w_i , the weight for the i th unlabeled example, is guaranteed to be non-negative. This is because $\sum_{k=1}^m \alpha_i^k + C\beta_i^k = 0$ and therefore $w_i = \max_k (\alpha_i^k + C\beta_i^k) \geq 0$; w_i is a measure of the failure of the algorithm on example x_i . Using the new weighting on the training examples, MCSSB learns a multi-class model that minimizes the loss on the weighted training examples, by adopting the sampling approach as described in the following: MCSSB samples s instances by replacement, with probability of each sample proportional to its weight. s sampled instances are passed to the weak learner to obtain a multi-class hypothesis. In our experiments, the number of sampled examples at each iteration is set as $s = \max(20, N/5)$. After creating a weak classifier at this round, MCSSB adds it to the current classifiers to reduce the value of the objective function.

For the experiments, we ran the algorithm with different numbers of iterations and find that both the objective function and the classification accuracy remains essentially the same after 50 iterations. We, therefore, set the number of iterations to be 50 to save the computational cost.

Algorithm 3 MCSSB: Multi-Class Semi-Supervised Boosting Algorithm

1: **Input:**

- D : The set of examples; the first N_l examples are labeled.
- s : the number of sampled examples from $(N - N_l)$ unlabeled examples
- T : the maximum number of iterations

2: Set $F(i) = 0, i = 1, \dots, |D|$

3: **repeat**

4: Compute α_i^k and β_i^k for every example as given in Equation 2.12.

5: Assign each unlabeled example x_i to class $k_i^* = \arg \min_k (\alpha_i^k + C\beta_i^k)$ and weight $w_i = \alpha_i^{k_i^*} + C\beta_i^{k_i^*}$

6: Sample s examples using a distribution that is proportional to w_i

7: Train a multi-class classifier $h(x)$ using the s samples examples

8: Predict h_i^k for all examples using $h(x)$, and compute α using Equation 4.14. Exit the loop if $\alpha \leq 0$.

8: $H(x) \leftarrow H(x) + \alpha h(x)$

9: **until** reach the maximum number of iterations

Theorem 4 shows that the proposed boosting algorithm reduces the objective function F exponentially. The proof of this theorem is provided in Appendix A.3.

Theorem 4. *The objective function after T iterations, denoted by F^T , is bounded as follows:*

$$F^T \leq F^0 \exp \left(- \sum_{t=1}^T \frac{(\sqrt{A_u^t + CA_l^t} - \sqrt{B_u^t + CB_l^t})^2}{F^{t-1}} \right) \quad (2.21)$$

where A_u , A_l , B_u and B_l are defined in Lemma 2.

2.4 Experiments

In this section, we present our empirical study on the classification data sets that were described in Chapter 7. We refer to the proposed semi-supervised multi-class boosting algorithm as **MCSSB**. In this study, we aim to show that (1) MCSSB can improve the performance of a given multi-class classifier with unlabeled examples, (2) MCSSB is more effective than the existing semi-supervised boosting algorithms, and (3) MCSSB is robust

to the model parameters and the number of labeled examples. It is important to note that it is not our intention to show that the proposed semi-supervised multi-class boosting algorithm always outperforms other semi-supervised learning algorithms. Instead, our objective is to demonstrate that the proposed semi-supervised boosting algorithm is able to effectively improve the accuracy of different supervised multi-class learning algorithms using the unlabeled examples. Hence, the empirical study is focused on a comparison with the existing semi-supervised boosting algorithms, rather than a wide range of semi-supervised learning algorithms.

2.4.1 Experimental Setup

For each classification data sets, described Section 1.6.1, we split the examples into 5 partitions, with one partition used for training and the others used for testing. In each experiment, we used a small percentage (between 2 to 10 percent) of training instances as labeled examples and the remaining instances as unlabeled examples. We applied the proposed algorithms and the baselines on the training examples to create a model and applied it on the test examples and computed the accuracy on the test examples. We repeated each experiment 10 times and reported the average.

We compare the proposed semi-supervised boosting algorithm to *ASSEMBLE*, a state-of-the-art semi-supervised boosting algorithm. The main reason for this choice was because Assemble utilizes boosting technique and can exploit an existing supervised learning technique. This makes the comparison fair and easy because it enables us to compare MC-SSB and Assemble with base classifiers that have different quality. Also notice that Assemble is a powerful semi-supervised learning technique that was the best semi-supervised algorithm among 34 participants in NIPS’2001 workshop competition "Unlabeled Data for Supervised Learning" [48].

Unlike the general setup introduced in 1.6.1, we used the test set for for mnist data set because of the huge size of the training set in mnist and the memory problem.

A Gaussian kernel is used as the measure for similarity in the standard *MCSSB* algorithm with kernel width set to be 15% of the range of the distance between examples for all the experiments, as suggested in [70]. To verify the importance of using the similarity measure in the semi-supervised boosting algorithm and direct formulation of multi-class problem, we use two other baselines: *MCSSB-Uniform* that uses similar similarity values for every pair of examples (i.e. $S_{ij} = 1, i, j = 1, \dots, N$) that can be considered MCSSB with a bad similarity measure, and *MCSSB-Absolute* that considers absolute similarity between an example and itself (i.e. $S_{ii} = 1, i = 1, \dots, N$) and absolute dissimilarity between two different examples (i.e. $S_{ij} = 0, i, j = 1, \dots, N \text{ \& } i \neq j$). *MCSSB-Absolute* can be considered MCSSB that only exploits the advantage of using a direct formulation of the multi-class problem.

We use decision tree with only two level of nodes, as the base classifier for all the methods in the standard setting . The combination parameter C is set to 10^4 in all experiments. To study the robustness of the proposed methods, we further investigate the effect of the depth of decision tree and combination parameter C on the performance of different methods in Sections 2.4.4 and 2.4.3 respectively.

2.4.2 Evaluation of Classification Performance

Figure 2.1 shows the result of different algorithms when the amount of labeled examples is changed from 2% to 10%. First, notice that MCSSB significantly improves the accuracy of decision tree for 5 out of 7 data sets. For data set 'Nursery', MCSSB performs worse than the base classifier and for data set 'Letter', the result of MCSSB is not much different than the base classifier. However, for both these cases, MCSSB-Absolute performs quite good that indicates the direct formulation of multi-class problem is useful and the bad

i.e. $0.15 \times (d_{\max} - d_{\min})$, where d_{\min} and d_{\max} are minimum and maximum distance between examples

Notice we also used neural network as another base classifier to evaluate the performance of our algorithm. Refer to [50] for the results on several benchmark datasets

performance is due to the utilization of a bad similarity matrix. Note that for several data sets, the improvement made by the MCSSB is dramatic. For instance, the classification accuracy of decision tree is improved from 33% to 48% for data set 'Pendigits', and from 24% to 43% for data set 'Optdigits' when there is 2% labeled examples; the classification accuracy of decision tree is improved from 13% to 17% for data set 'Isolet', and from 46% to 49% for data set 'Protein' when there is 8% labeled examples.

Second, when compared to ASSEMBLE, we found that the proposed algorithm significantly outperforms ASSEMBLE for all the data sets. More interestingly, Assemble reduces the performance of the base classifier for most data sets that indicates the usage of pseudo-labels can produce misleading results. The key differences between MCSSB and ASSEMBLE is that MCSSB is not only specially designed for multi-class classification, it does not solely rely on the pseudo-labels obtained in the iterations of boosting algorithm. Thus, the success of MCSSB indicates the importance of designing semi-supervised learning algorithms for multi-class problems.

Third, to verify that the outstanding performance of MCSSB is related to the direct formulation of multi-class problem and the usage of similarity measure in the boosting algorithm, we examine the results of MCSSB-Uniform and MCSSB-Absolute. Because MCSSB-Uniform does not utilize an appropriate similarity measure, it performs very poorly that emphasizes our effective approach in utilizing the similarity measure in the boosting algorithm. On the other hand, MCSSB-Absolute is the second best method after MCSSB. Because MCSSB-Absolute does not utilize any similarity measure among examples, we believe that this superior performance is due to our approach in direct formulation of multi-class problem. It is interesting to note that the performance of MCSSB-Absolute on the 'Nursery' and 'Letter' data sets is better than other methods including MCSSB that indicated the sensitivity of the proposed method to the choice of similarity method.

And finally, notice that as the number of labeled examples increases, the performance of different methods improves. However MCSSB keeps its superiority for most of the cases

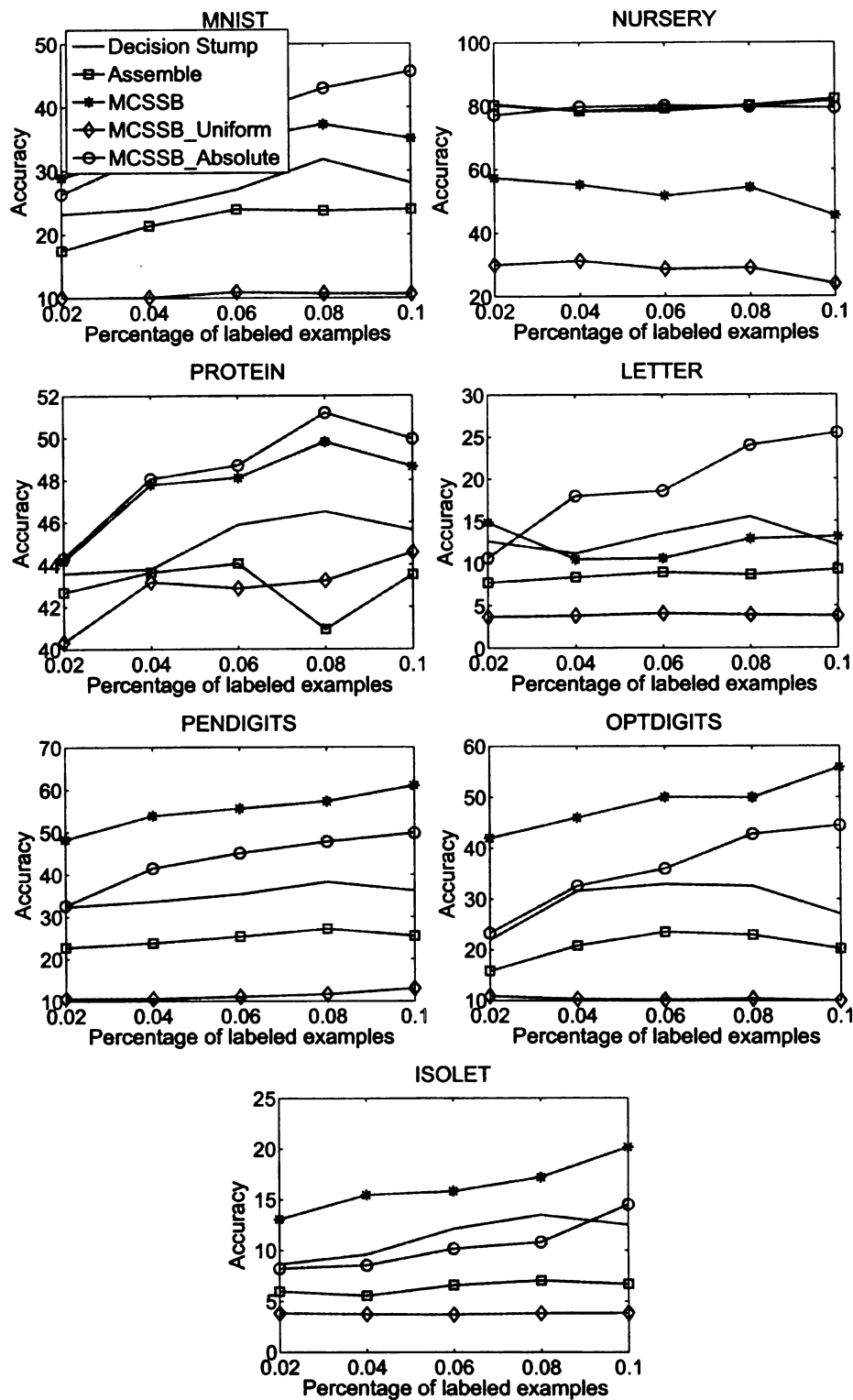


Figure 2.1: The error rates of different methods with different amount of labeled examples.

when compared to both the base classifier and the ASSEMBLE algorithm. We also observe that overall ASSEMBLE is unable to make improvement over the base classifier regardless of the number of labeled examples. These results indicate the challenge in developing boosting algorithms for semi-supervised multi-class learning. Compared to ASSEMBLE that relies on the classification confidence to decide the pseudo labels for unlabeled examples, MCSSB is more reliable since it exploits both the classification confidence and similarities among examples when determining the pseudo labels.

2.4.3 Sensitivity to the Combination Parameter C

Figure 2.2 shows the performance of MCSSB when the combination parameter C changes from 1 to 10^{10} . It is clear that for large values of C , MCSSB is very stable. Notice the improvement of MCSSB on the base classifier for dataset 'Protein' is very marginal for some values of C . However if you look at Figure 2.1, you will notice that the result of MCSSB for larger amount of labeled data (as large as 4%) is significant for this data set and not sensitive to the small changes of parameters C . We conclude that MCSSB is very robust to the choice of parameter C .

2.4.4 Sensitivity to Base Classifier

In this section, we focus on examining the sensitivity of MCSSB to the complexity of base classifiers. This will allow us to understand the behavior of the proposed semi-supervised boosting algorithm for both weak classifiers and strong classifiers. To this end, we use decision tree with varying number of levels as the base classifier. We used decision tree with only one node (decision stump) up to fully-grown decision trees and plot the performance result of different methods. Figure 2.3 shows the classification accuracy of Tree, ASSEMBLE and MCSSB when we vary the number of levels in decision tree. Notice that in each case, the maximum number of levels in the plot for each data set is set to the fully grown tree for that data set. It is not surprising that overall the classification accuracy is improved

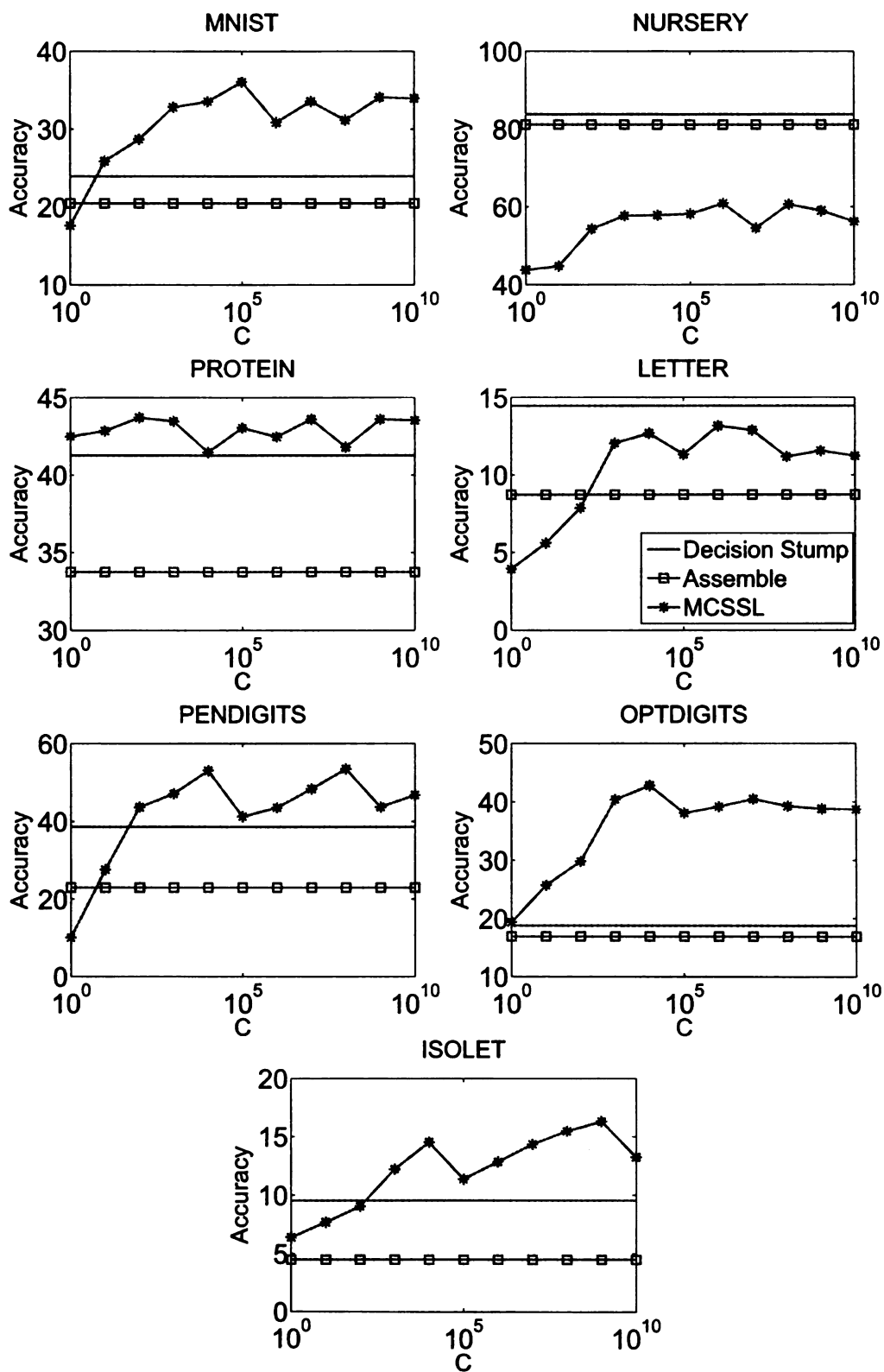


Figure 2.2: The error rates of MCSSB with different C (2% of labeled).

with increasing number of levels in decision tree for most data sets. We also observe that MCSSB is more effective than ASSEMBLE for decision trees with different complexity and regardless of quality of the base classifier, ASSEMBLE is not able to improve the performance of the supervised classifier by utilizing unlabeled examples. Notice that for some data sets, e.g. 'Protein' data set, the performance decreases as the depth of tree increases. This is because, unlike other data sets, 'Protein' has only three classes and large tree can lead to overfitting.

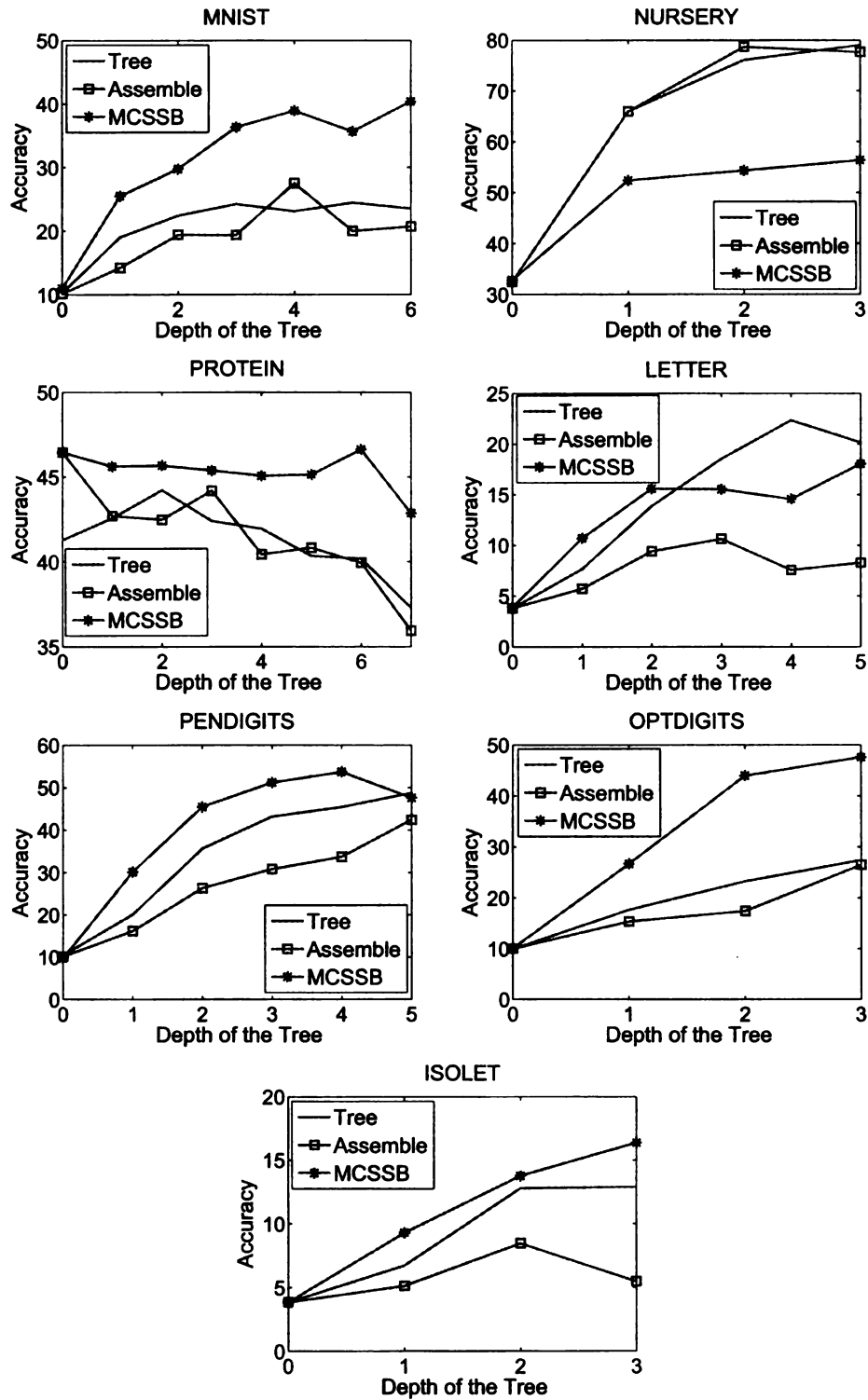


Figure 2.3: The error rates of MCSSB with decision tree with different depth as the weak learner. 2% of training examples are labeled in all the experiments.

Chapter 3

Optimizing NDCG Measure by Boosting

Learning to rank is a relatively new field in machine learning. It aims to learn a ranking function from training examples with relevancy judgements. The learning to rank algorithms are often evaluated using information retrieval measures, such as Normalized Discounted Cumulative Gain (NDCG) [14] and Mean Average Precision (MAP) [13]. Until recently, most learning to rank algorithms were not able to directly optimize a loss function related to the IR evaluation measures, such as NDCG and MAP. The main difficulty in direct optimization of these measures is that they are non-continuous and non-differentiable. In this chapter, we discuss how boosting can be applied to optimize Normalized Discounted Cumulative Gain (NDCG) which is the most commonly used multi-level evaluation measure for learning to rank. We start with a detailed description of AdaRank [42], one of the first algorithms designed to directly maximize IR measures. We further develop a learning to rank algorithm, termed NDCG_Boost, for optimizing NDCG metric. Unlike AdaRank that weights all the documents related to each query equally when optimizing the NDCG measure, NDCG_Boost weights individual documents differently even if they are all related to the same query, leading to more effective optimization of the NDCG measure. In order to deal with the non-smooth nature of the NDCG measure, in the NDCG_Boost algorithm, we propose to optimize the expectation of NDCG over the distribution induced

by a ranking function. We then present a relaxation strategy that approximates the average of NDCG value, and an optimization strategy to make the computation efficient. Extensive experiments show that the proposed algorithm outperforms state-of-the-art ranking algorithms on several benchmark data sets.

3.1 Introduction

Learning to rank has attracted many machine learning researchers in the last decade because of its growing importance in the areas like information retrieval (IR) and recommender systems. Three types of learning to rank algorithms can be found in the literature.

- **Pointwise approaches:** As the simplest form, these approaches [15, 16] treat ranking as a classification or regression problem that learns a ranking function in order to fit the relevance judgments for given retrieved documents [16, 17]. However classification and regression may not be the best for the task of ranking. This is because (i) classification problems are usually associated with unordered class labels where there is an intrinsic order among the levels of relevance judgments provided by the user, and (ii) the target variables in regression problems are assumed to be numerical values while the relevance judgments are only ordinary variables.
- **Pairwise approaches:** These approaches are motivated by the fact that the relevancy scores in ranking are relative to each other. This group considers the pairs of documents as independent variables and learns a classification (regression) model to correctly order the training pairs [18–23], namely document d_a is ranked above d_b if the relevance score of d_a is larger than d_b . One major problem with the pairwise approaches is that they assume pairs of documents are independent random variables, which is often violated in real world applications. .

- **Listwise approaches:** The listwise approaches are motivated by this observation that most evaluation metrics of information retrieval measure the ranking quality for individual queries, not documents. These approaches treat the ranking list of documents for every query as a training instance [13, 24–29], either by direct optimization of an information retrieval evaluation measure [13, 25, 28, 29] or by optimizing a listwise loss function [24, 26, 27]. Empirical studies have shown that the listwise approaches are more effective than both pointwise and listwise approaches because they utilize the query-document group structure which is a unique and useful characteristic in ranking.

The main difficulty in optimizing the listwise loss functions is that they are non-continuous and non-differentiable. This is because these loss functions measure the retrieval performance based on the ranking list of documents induced by the ranking function, and therefore their dependence on ranking functions is implicit. Given that classification is a well-studied subject in machine learning, the research question is whether it is possible to design a boosting algorithm that utilizes a classification algorithm to optimize an information retrieval measure such as NDCG. The easiest way to design such a boosting algorithm is the approach taken by Xu et al. in the design of AdaRank [42]. In each trial of a boosting algorithm, AdaRank re-weights the queries based on their NDCG values (compared to AdaBoost that re-weights the examples based on their confidence in prediction). As we see in more details in Section 3.3.2, AdaRank treats all the documents related to each query equally when trying to improve the NDCG metric, which could significantly limits the choice of ranking functions for optimizing the NDCG metric. In this chapter, we introduce a better boosting algorithm for optimizing NDCG metric that weights documents differently even if they are associated with the same query. In each iteration, the boosting algorithm provides a weighting as well as binary class assignments for given documents; the weak learner constructs a binary classifier from the weighted documents that are labeled

It is important to distinguish the binary class assignment from the relevance judgments for documents

by the boosting algorithm.

3.2 Related Work

We focus on reviewing the listwise approaches that are closely related to the theme of this chapter. The listwise approaches can be classified into two categories. The first group of approaches directly optimizes the IR evaluation metrics. Most IR evaluation metrics, however, depend on the sorted order of documents, and are non-convex in the target ranking function. To avoid the computational difficulty, these approaches either approximate the metrics with some convex functions or deploy methods (e.g., genetic algorithm [71]) for non-convex optimization. In [25], the authors introduced LambdaRank that addresses the difficulty in optimizing IR metrics by defining a virtual gradient on each document after the sorting. While [25] provided a simple test to determine if there exists an implicit cost function for the virtual gradient, theoretical justification for the relation between the implicit cost function and the IR evaluation metric is incomplete. This may partially explain why LambdaRank performs very poor when compared to MCRank [16], a simple adjustment of classification for ranking (a pointwise approach). The authors of MCRank paper even claimed that a boosting model for regression produces better results than LambdaRank. Volkovs and Zemel [29] proposed optimizing the expectation of IR measures to overcome the sorting problem, similar to the approach taken in this paper. However they use monte carlo sampling to address the intractable task of computing the expectation in the permutation space which could be a bad approximation for the queries with large number of documents. AdaRank [42], as was described earlier in this chapter, uses boosting to optimize NDCG, similar to our optimization strategy. However they deploy heuristics to embed the IR evaluation metrics in computing the weights of queries and the importance of weak rankers; i.e. it uses NDCG value of each query in the current iteration as the weight for that query in constructing the weak ranker (the documents of each query have

similar weight). This is unlike our approach that the contribution of each single document to the final NDCG score is considered. Moreover, unlike our method, the convergence of AdaRank is conditional and not guaranteed. Sun et al. [72] reduced the ranking, as measured by NDCG, to pairwise classification and applied alternating optimization strategy to address the sorting problem by fixing the rank position in getting the derivative. SVM-MAP [13] relaxes the MAP metric by incorporating it into the constraints of SVM. Since SVM-MAP is designed to optimize MAP, it only considers the binary relevancy and cannot be applied to the data sets that have more than two levels of relevance judgements.

The second group of listwise algorithms defines a listwise loss function as an indirect way to optimize the IR evaluation metrics. RankCosine [24] uses cosine similarity between the ranking list and the ground truth as a query level loss function. ListNet [26] adopts the KL divergence for loss function by defining a probabilistic distribution in the space of permutation for learning to rank. FRank [22] uses a new loss function called fidelity loss on the probability framework introduced in ListNet. ListMLE [27] employs the likelihood loss as the surrogate for the IR evaluation metrics. The main problem with this group of approaches is that the connection between the listwise loss function and the targeted IR evaluation metric is unclear, and therefore optimizing the listwise loss function may not necessarily result in the optimization of the IR metrics.

3.3 Optimizing NDCG Measure

3.3.1 Notation

Assume that we have a collection of n queries for training, denoted by $\mathcal{Q} = \{q^1, \dots, q^n\}$. For each query q^k , we have a collection of m_k documents $\mathcal{D}^k = \{d_i^k, i = 1, \dots, m_k\}$, whose relevance to q^k is given by a vector $r^k = (r_1^k, \dots, r_{m_k}^k) \in \mathbb{Z}^{m_k}$. We denote by $F(d, q)$ the ranking function that takes a document-query pair (d, q) and outputs a real number score, and by j_i^k the rank of document d_i^k within the collection \mathcal{D}^k for query q^k .

The NDCG value for ranking function $F(d, q)$ is then computed as following:

$$\mathcal{L}(Q, F) = \frac{1}{n} \sum_{k=1}^n \frac{1}{Z_k} \sum_{i=1}^{m_k} \frac{2^{r_i^k} - 1}{\log(1 + j_i^k)} \quad (3.1)$$

where Z_k is the normalization factor [14]. NDCG is usually truncated at a particular rank level (e.g. the first 10 retrieved documents) to emphasize the importance of the first retrieved documents.

3.3.2 AdaRank Algorithm

The easiest way to design a boosting algorithm for optimizing a given IR evaluation measure is what AdaRank algorithm [42] performs. AdaRank uses an exponential loss function similar to AdaBoost. However, unlike the loss function of AdaBoost which is constructed based on the classification margin, AdaRank utilizes information retrieval measures such as NDCG to construct the exponential loss. To optimize NDCG, for example, AdaRank uses the following exponential loss function:

$$\sum_{k=1}^n \exp(-\mathcal{L}(q_k, F))$$

where $\mathcal{L}(q_k, F)$ is the NDCG value for query k when ranking the documents for query q_k by function F . The steps of AdaRank are given in Algorithm 4. In each iteration, AdaRank finds a weak ranker f_t that maximizes quantity η_t at Step 4, i.e. NDCG weighted by p . Then, it computes the combination weight for f_t and adds it to the current set of classifiers in Steps 5 and 6 respectively. The authors of AdaRank paper [42] suggest using the ranking features (e.g. BM25) as the weak ranker. However, a (multi-class) classifier can also be used as the weak ranker. To construct a classifier that maximizes η_t , AdaRank distributes the weight $p_t(k)$ to all documents of query k equally, and constructs a classifier based on the documents that are sampled according to the weights. To redistributes the weights to

Algorithm 4 AdaRank Algorithm

1: **Input:**

- $\mathcal{Q} = \{q^1, \dots, q^n\}$: The set of queries
- $\mathcal{D}^k = \{(d_i^k, r_i^k), i = 1, \dots, m_k\}$: The set of documents and their relevancy scores for query q^k .

2: Initialize $p_1(q_k) = 1/n, k = 1, \dots, n$

3: **repeat**

4: Find f_t by maximizing weighted NDCG; i.e. $\eta_t = \sum_{i=1}^k p_t(q_k) \mathcal{L}(q_k, F)$

5: Compute $\alpha_t = \frac{1}{2} \ln(\frac{\eta_t}{1-\eta_t})$

6: Compute $F(d_i^k) = \sum_{l=1}^t \alpha_l f_l(d_i^k), k = 1, \dots, n, i = 1, \dots, m_k$

7: Compute the new weighting $p_{t+1}(q_k) = \frac{\exp(-\mathcal{L}(Q_k, F))}{\sum_{k=1}^n \exp(-\mathcal{L}(Q_k, F))}$

8: **until** reach the maximum number of iterations

instanced, AdaRank increases the weights of difficult queries (e.g. those that have small NDCG) and decreases the weights of easy queries (e.g. those that have large NDCG) at Step 7.

As it is obvious from the steps of AdaRank algorithm, it gives the same weights to the documents of each query, leading to a suboptimal performance. However, since a pointwise weak learner (multi-class classifier) is often utilized in a boosting algorithm to maximize NDCG, it is advantageous to allow every document to contribute differently to the final NDCG value. Moreover, although NDCG works in query level, not all documents have similar contribution in improving the NDCG value at each stage of the algorithm. These observations motivated us to develop NDCG_Boost algorithm that considers the contribution of every single document in the iterations of the boosting algorithm to maximize NDCG.

3.3.3 A Probabilistic Framework

One of the main challenges faced by optimizing the NDCG metric defined in Equation (3.1) is that the dependence of document ranks (i.e., j_i^k) on the ranking function $F(d, q)$ is not explicitly expressed, which makes it computationally challenging. To address this problem, we consider the expectation of $\mathcal{L}(Q, F)$ over all the possible rankings induced by

the **ranking** function $F(d, q)$, i.e.,

$$\begin{aligned}\bar{\mathcal{L}}(Q, F) &= \frac{1}{n} \sum_{k=1}^n \frac{1}{Z_k} \sum_{i=1}^{m_k} \left\langle \frac{2^{r_i^k} - 1}{\log(1 + j_i^k)} \right\rangle_F \\ &= \frac{1}{n} \sum_{k=1}^n \frac{1}{Z_k} \sum_{i=1}^{m_k} \sum_{\pi^k \in S_{m_k}} \Pr(\pi^k | F, q^k) \frac{2^{r_i^k} - 1}{\log(1 + \pi^k(i))}\end{aligned}\quad (3.2)$$

where S_{m_k} stands for the group of permutations of m_k documents, and π^k is an instance of permutation (or ranking). Notation $\pi^k(i)$ stands for the rank position of the i th document by π^k . To this end, we first utilize the result in the following lemma to approximate the expectation of $1/\log(1 + \pi^k(i))$ by the expectation of $\pi^k(i)$.

Lemma 3. *For any distribution $\Pr(\pi | F, q)$, the inequality $\bar{\mathcal{L}}(Q, F) \geq \bar{\mathcal{H}}(Q, F)$ holds where*

$$\bar{\mathcal{H}}(Q, F) = \frac{1}{n} \sum_{k=1}^n \frac{1}{Z_k} \sum_{i=1}^{m_k} \frac{2^{r_i^k} - 1}{\log(1 + \langle \pi^k(i) \rangle_F)} \quad (3.3)$$

Proof. The proof follows from the fact that (a) $1/x$ is a convex function when $x > 0$ and therefore $\langle 1/\log(1 + x) \rangle \geq 1/\langle \log(1 + x) \rangle$; (b) $\log(1 + x)$ is a concave function, and therefore $\langle \log(1 + x) \rangle \leq \log(1 + \langle x \rangle)$. Combining these two factors together, we have the result stated in the lemma. \square

Given $\bar{\mathcal{H}}(Q, F)$ provides a lower bound for $\bar{\mathcal{L}}(Q, F)$, in order to maximize $\bar{\mathcal{L}}(Q, F)$, we could alternatively maximize $\bar{\mathcal{H}}(Q, F)$, which is substantially simpler than $\bar{\mathcal{L}}(Q, F)$. In the next step of simplification, we rewrite $\pi^k(i)$ as

$$\pi^k(i) = 1 + \sum_{j=1}^{m_k} I(\pi^k(i) > \pi^k(j)) \quad (3.4)$$

where $I(x)$ outputs 1 when x is true and zero otherwise. Hence, $\langle \pi^k(i) \rangle$ is written as

$$\langle \pi^k(i) \rangle = 1 + \sum_{j=1}^{m_k} \langle I(\pi^k(i) > \pi^k(j)) \rangle = 1 + \sum_{j=1}^{m_k} \Pr(\pi^k(i) > \pi^k(j)) \quad (3.5)$$

As a result, to optimize $\bar{\mathcal{H}}(Q, F)$, we only need to define $\Pr(\pi^k(i) > \pi^k(j))$, i.e., the marginal distribution for document d_j^k to be ranked before document d_i^k . In the next section, we will discuss how to define a probability model for $\Pr(\pi^k|F, q^k)$, and derive pairwise ranking probability $\Pr(\pi^k(i) > \pi^k(j))$ from distribution $\Pr(\pi^k|F, q^k)$.

3.3.4 Objective Function

We model $\Pr(\pi^k|F, q^k)$ as follows

$$\begin{aligned} \Pr(\pi^k|F, q^k) &= \frac{1}{Z(F, q^k)} \exp \left(\sum_{i=1}^{m_k} \sum_{j: \pi^k(j) > \pi^k(i)} (F(d_i^k, q^k) - F(d_j^k, q^k)) \right) \\ &= \frac{1}{Z(F, q^k)} \exp \left(\sum_{i=1}^{m_k} (m_k - 2\pi^k(i) + 1) F(d_i^k, q^k) \right) \end{aligned} \quad (3.6)$$

where $Z(F, q^k)$ is the partition function that ensures the sum of probability is one. Equation (3.6) models each pair (d_i^k, d_j^k) of the ranking list π^k by the factor $\exp(F(d_i^k, q^k) - F(d_j^k, q^k))$ if d_i^k is ranked before d_j^k (i.e., $\pi^k(d_i^k) < \pi^k(d_j^k)$) and vice versa. This modeling choice is consistent with the idea of ranking the documents with largest scores first; intuitively, the more documents in a permutation are in the decreasing order of score, the bigger the probability of the permutation is. Using Equation (3.6) for $\Pr(\pi^k|F, q^k)$, we have $\bar{\mathcal{H}}(Q, F)$ expressed in terms of ranking function F . By maximizing $\bar{\mathcal{H}}(Q, F)$ over F , we could find the optimal solution for ranking function F .

As indicated by Equation (3.5), we only need to compute the marginal distribution $\Pr(\pi^k(i) > \pi^k(j))$. To approximate $\Pr(\pi^k(i) > \pi^k(j))$, we divide the group of permutation S_{m_k} into two sets: $G_a^k(i, j) = \{\pi^k | \pi^k(i) > \pi^k(j)\}$ and $G_b^k(i, j) = \{\pi^k | \pi^k(i) < \pi^k(j)\}$.

$\pi^k(j)\}$. Notice that there is a one-to-one mapping between these two sets; namely for any ranking $\pi^k \in G_a^k(i, j)$, we could create a corresponding ranking $\pi^k \in G_b^k(i, j)$ by switching the rankings of document d_i^k and d_j^k and vice versa. The following lemma allows us to bound the marginal distribution $\Pr(\pi^k(i) > \pi^k(j))$. The proof of this lemma is provided in Appendix A.5.

Lemma 4. *If $F(d_i^k, q^k) > F(d_j^k, q^k)$, we have*

$$\Pr(\pi^k(i) > \pi^k(j)) \leq \frac{1}{1 + \exp \left[2(F(d_i^k, q^k) - F(d_j^k, q^k)) \right]} \quad (3.7)$$

This lemma indicates that we could approximate $\Pr(\pi^k(i) > \pi^k(j))$ by a simple logistic model. The idea of using logistic model for $\Pr(\pi^k(i) > \pi^k(j))$ is not new in learning to rank [20, 22]; however it has been taken for granted and no justification has been provided in using it for learning to rank. Using the logistic model approximation introduced in Lemma 4, we now have $\langle \pi^k(i) \rangle$ written as

$$\langle \pi^k(i) \rangle \approx 1 + \sum_{j=1}^{m_k} \frac{1}{1 + \exp \left[2(F(d_i^k, q^k) - F(d_j^k, q^k)) \right]} \quad (3.8)$$

To simplify our notation, we define $F_i^k = 2F(d_i^k, q^k)$, and rewrite the above expression as

$$\langle \pi^k(i) \rangle = 1 + \sum_{j=1}^{m_k} \Pr(\pi^k(i) > \pi^k(j)) \approx 1 + \sum_{j=1}^{m_k} \frac{1}{1 + \exp(F_i^k - F_j^k)}$$

Using the above approximation for $\langle \pi^k(i) \rangle$, we have $\bar{\mathcal{H}}$ in Equation (3.3) written as

$$\bar{\mathcal{H}}(Q, F) \approx \frac{1}{n} \sum_{k=1}^n \frac{1}{Z_k} \sum_{i=1}^{m_k} \frac{2^{r_i^k} - 1}{\log(2 + A_i^k)} \quad (3.9)$$

where

$$A_i^k = \sum_{j=1}^{m_k} \frac{I(j \neq i)}{1 + \exp(F_i^k - F_j^k)} \quad (3.10)$$

We define the following proposition to further simplify the objective function:

Proposition 1.

$$\frac{1}{\log(2 + A_i^k)} \geq \frac{1}{\log(2)} - \frac{A_i^k}{2[\log(2)]^2}$$

The proof is due to the Taylor expansion of convex function $1/\log(2 + x)$, $x > -1$ around $x = 0$ noting that $A_i^k > 0$ (the proof of convexity of $1/\log(1 + x)$ is given in Lemma 3) and is provided in Appendix A.6. By plugging the result of this proposition to the objective function in Equation (3.9), the new objective is to minimize the following quantity:

$$\bar{\mathcal{M}}(Q, F) \approx \frac{1}{n} \sum_{k=1}^n \frac{1}{Z_k} \sum_{i=1}^{m_k} (2^{r_i^k} - 1) A_i^k \quad (3.11)$$

The objective function in Equation (3.11) is explicitly related to F via term A_i^k . In the next section, we aim to derive an algorithm that learns an effective ranking function by efficiently minimizing $\bar{\mathcal{M}}$. It is also important to note that although $\bar{\mathcal{M}}$ is no longer a rigorous lower bound for the original objective function $\bar{\mathcal{L}}$, our empirical study shows that this approximation is very effective in identifying the appropriate ranking function from the training data.

3.3.5 Algorithm

To minimize $\bar{\mathcal{M}}(Q, F)$ in Equation (3.11), we employ the boosting strategy [38] that iteratively updates the solution for F . Let F_i^k denote the value obtained so far for document

d_i^k . To improve NDCG, following the idea of Adaboost, we restrict the new ranking value for document d_i^k , denoted by \tilde{F}_i^k , is updated as to the following form:

$$\tilde{F}_i^k = F_i^k + \alpha f_i^k \quad (3.12)$$

where $\alpha > 0$ is the combination weight and $f_i^k = f(d_i^k, q^k) \in \{0, 1\}$ is a binary value. Note that in the above, we assume the ranking function $F(d, q)$ is updated iteratively with an addition of binary classification function $f(d, q)$, which leads to efficient computation as well as effective exploitation of the existing algorithms for data classification. To construct a lower bound for $\bar{\mathcal{M}}(Q, F)$, we first handle the expression $[1 + \exp(F_i^k - F_j^k)]^{-1}$, summarized by the following proposition.

Proposition 2.

$$\frac{1}{1 + \exp(\tilde{F}_i^k - \tilde{F}_j^k)} \leq \frac{1}{1 + \exp(F_i^k - F_j^k)} + \gamma_{i,j}^k [\exp(\alpha(f_j^k - f_i^k)) - 1] \quad (3.13)$$

where

$$\gamma_{i,j}^k = \frac{\exp(F_i^k - F_j^k)}{(1 + \exp(F_i^k - F_j^k))^2} \quad (3.14)$$

The proof of this proposition can be found in Appendix A.4. This proposition separates the term related to F_i^k from that related to αf_i^k in Equation (3.11), and shows how the new weak ranker (i.e., the binary classification function $f(d, q)$) will affect the current ranking function $F(d, q)$. Using the above proposition, we can derive the upper bound for $\bar{\mathcal{M}}$ (Theorem 5) as well as a closed form solution for α given the solution for F (Theorem 6).

Theorem 5. Given the solution for binary classifier f_i^d , the optimal α that minimizes the

objective function in Equation (3.11) is

$$\alpha = \frac{1}{2} \log \left(\frac{\sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k I(f_j^k < f_i^k)}{\sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k I(f_j^k > f_i^k)} \right) \quad (3.15)$$

where $\theta_{i,j}^k = \gamma_{i,j}^k I(j \neq i)$.

Remark: Notice that in order to have this boosting algorithm continue the iterations, the weak learner needs to produce models better than random guessing in the following sense.

Writing α in the following form

$$\alpha = \frac{1}{2} \log \left(\frac{1 - \epsilon}{\epsilon} \right) \quad (3.16)$$

where

$$\epsilon = \frac{\sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k I(f_j^k > f_i^k)}{\sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k (I(f_j^k > f_i^k) + I(f_j^k < f_i^k))}$$

, we mean $\epsilon \leq 0.5$ by random guessing.

Theorem 6.

$$\bar{\mathcal{M}}(Q, \tilde{F}) \leq \bar{\mathcal{M}}(Q, F) + \gamma(\alpha) + \frac{\exp(3\alpha) - 1}{3} \sum_{k=1}^n \sum_{i=1}^{m_k} f_i^k \left(\sum_{j=1}^{m_k} \frac{2^{r_i^k} - 2^{r_j^k}}{Z_k} \theta_{i,j}^k \right)$$

where $\gamma(\alpha)$ is only a function of α with $\gamma(0) = 0$.

The proofs of these theorems are provided in Appendix A.6 and Appendix A.7 respectively. Note that the bound provided by Theorem 6 is tight because by setting $\alpha = 0$, the inequality reduces to equality resulting $\bar{\mathcal{M}}(Q, \tilde{F}) = \bar{\mathcal{M}}(Q, F)$. The importance of this theorem is that the optimal solution for f_i^k can be found without knowing the solution for α .

Algorithm 5 NDCG_Boost: A Boosting Algorithm for Maximizing NDCG

- 1: Initialize $F(d_i^k) = 0$ for all documents
- 2: **repeat**
- 3: Compute $\theta_{i,j}^k = \gamma_{i,j}^k I(j \neq i)$ for all document pairs of each query. $\gamma_{i,j}^k$ is given in Eq. (3.14).
- 4: Compute the weight for each document as

$$w_i^k = \sum_{j=1}^{m_k} \frac{2^{r_i^k} - 2^{r_j^k}}{Z_k} \theta_{i,j}^k \quad (3.17)$$

- 5: Assign each document the following class label $y_i^k = \text{sign}(w_i^k)$.
- 6: Train a classifier $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \{0, 1\}$ that maximizes the following quantity

$$\eta_t = \sum_{k=1}^n \sum_{i=1}^{m_k} |w_i^k| f(d_i^k) y_i^k \quad (3.18)$$

- 7: Predict f_i for all documents in $\{D^k, i = 1, \dots, n\}$
 - 8: Compute the combination weight α as provided in Equation (3.15).
 - 9: Update the ranking function as $F_i^k \leftarrow F_i^k + \alpha f_i^k$.
 - 10: **until** reach the maximum number of iterations
-

Algorithm 5 summarizes the boosting algorithm in minimizing the objective function in Equation (3.11). In each iteration, it computes θ_{ij}^k for every pair of documents of query k . θ_{ij}^k can be considered a measure of how close the rank position of documents d_i^k and d_j^k are when they are sorted by function F . The algorithm computes w_i^k , a weight for each document, which is the summary information of document d_i^k when its position and relevancy score compared to all other documents of the same query. w_i^k can be positive or negative. A positive w_i^k indicates that the ranking position of d_i^k induced by the current ranking function F is less than its true rank position, and a negative weight w_i^k shows that ranking position of d_i^k induced by the current F is greater than its true rank position. The magnitude of w_i^k shows how much the corresponding document is misplaced in the ranking. In other words, it shows the importance of correct ranking position of document d_i^k in terms of the value of NDCG. Using these information, the algorithm finds out the most difficult

Notice that we use $F(d_i^k)$ instead of $F(d_i^k, q^k)$ to simplify the notation in the algorithm.

documents and the relevancy direction of their importance at the current iteration. Using these information, NDCG_Boost maximizes η_t as given by Equation (3.18) which can be considered as some sort of classification accuracy. It uses sampling strategy in order to maximize η_t because most binary classifiers do not support the weighted training set; that is, it first samples the documents according to $|w_i^k|$ and then constructs a binary classifier with the sampled documents. After learning the new binary model at Step 6, the algorithm evaluates its success in improving the value of NDCG in Step 7 and 8 and adds it to the current set of binary models (the mixed strategy over binary models) at Step 9.

The following theorem shows that the proposed boosting algorithm reduces the objective function M exponentially.

Theorem 7. *The objective function after T iterations, denoted by $\bar{\mathcal{M}}^T$, is bounded as follows:*

$$\bar{\mathcal{M}}^T \leq \bar{\mathcal{M}}^0 \exp\left(-\sum_{t=1}^T \frac{(\sqrt{\alpha_1^t} - \sqrt{\alpha_2^t})^2}{\bar{\mathcal{M}}^{t-1}}\right)$$

where α_1 and α_2 are defined as follows.

$$\alpha_1 = \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k I(f_j^k < f_i^k) \quad , \quad \alpha_2 = \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k I(f_j^k > f_i^k) \quad (3.19)$$

The proof is provided in Appendix A.8.

3.4 Experiments

To study the performance of NDCG_Boost we use the latest version (version 3.0) of LETOR package provided by Microsoft Research Asia [56], which has been described in Chapter 1. Besides a number of benchmark data data, LETOR package also includes multiple state-of-the-art baselines and evaluation tools for research on learning to rank.

3.4.1 Experimental setup

A number of state-of-the-art learning to rank algorithms are provided in the LETOR package, including some of the most well-known learning to rank algorithms from each category (pointwise, pairwise and listwise). These baselines will be used to study the performance of NDCG_Boost. Here is the list of these baselines (the details can be found in the LETOR web page):

Regression: This is a pointwise approach that applies a linear regression to a ranking problem. It is used as a reference point.

RankSVM: RankSVM is a pairwise approach that applies Support Vector Machine [18] to the ranking problem.

FRank: FRank is a pairwise approach. It uses a probability model similar to RankNet [20] for the relative rank position of two documents, with a novel loss function called Fidelity loss function [22]. TSai et al. [22] showed that FRank performs significantly better than RankNet.

ListNet: ListNet is a listwise learning to rank algorithm [26]. It uses cross-entropy loss as its listwise loss function.

AdaRank_NDCG: This is a listwise boosting algorithm that incorporates NDCG in computing the weights for both queries and the combination of weak ranking hypotheses [42].

SVM_MAP: SVM_MAP is a support vector machine with MAP measure as the target objective function. It is a listwise approach [13].

While the validation set is used in finding the best set of parameters in the baselines in LETOR, it is not used for NDCG_Boost in our experiments. For NDCG_Boost, we set the maximum number of iteration to 100 and use decision stump as the weak ranker.

3.4.2 Results

Figure 3.1 provides the the average results of five folds for different learning to rank algorithms in terms of NDCG @ each of the first 10 truncation level on the LETOR data sets. Notice that the performance of algorithms in comparison varies from one data set to another; however NDCG_Boost performs almost always the best. We would like to point out a few statistics; On OHSUMED data set, NDCG_Boost performs 0.50 at $NDCG@3$, a 4% increase in performance, compared to FRANK, the second best algorithm. On TD2003 data set, this value for NDCG_Boost is 0.375 that shows a 10% increase, compared with RankSVM (0.34), the second best method. On HP2004 data set, NDCG_Boost performs 0.80 at $NDCG@3$, compared to 0.75 of SVM_MAP, the second best method, which indicates a 6% increase. Moreover, among all the methods in comparison, NDCG_Boost appears to be the most stable method across all the data sets. For example, FRank, which performs well in OHSUMED and TD2004 data sets, yields a poor performance on TD2003, HP2003 and HP 2004. Similarly, AdaRank_NDCG achieves a decent performance on OHSUMED data set, but fails to deliver accurate ranking results on TD2003, HP2003 and NP2003. In fact, both AdaRank_NDCG and FRank perform even worse than the simple Regression approach on TD2003, which further indicates their instability. As another example, ListNet and RankSVM, which perform well on TD2003 are not competitive to NDCG_boost on OHSUMED and TD2004 data sets.

NDCG is commonly measured at the first few retrieved documents to emphasize their importance.

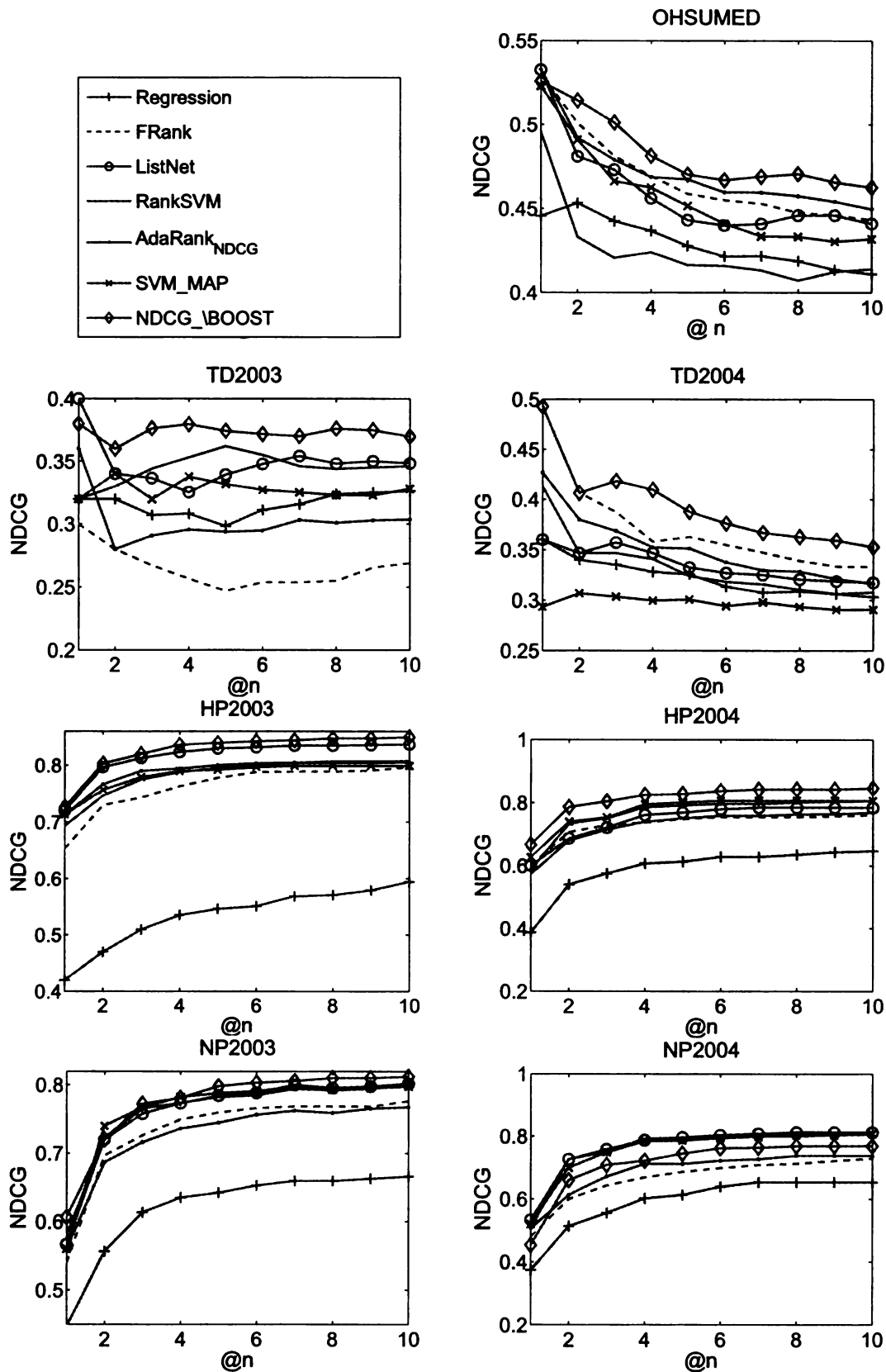


Figure 3.1: The experimental results in terms of NDCG for Letor 3.0 data sets

Chapter 4

Ranking Refinement by Boosting

In this chapter, we consider the problem of improving the accuracy of an existing ranking function with a small set of labeled instances. We are particularly interested in learning a better ranking function using two complementary sources of information, ranking information given by the existing ranking function (i.e., the base ranker) and that obtained from user feedback. We call this problem **ranking refinement**. Ranking refinement is very important in information retrieval where feedbacks are gradually collected. The key challenge in combining the two sources of information arises from the fact that the ranking information presented by the base ranker tends to be imperfect and the ranking information obtained from users' feedbacks tends to be noisy. We develop an objective function based on the pairwise approach for this problem and utilize the boosting technique to optimize it. Our empirical study shows that the proposed boosting algorithm is effective for ranking refinement, and furthermore it significantly outperforms the baseline algorithms that incorporate the outputs from the base ranker as an additional feature.

4.1 Introduction

Most research in learning to rank is conducted in the supervised fashion, in which a ranking function is learned from a given set of training instances. The drawback with the supervised

approach is that they tend to fail when the number of training instances is small. In several real-world applications, in addition to the labeled training instances, a *base ranker* is available that can be used to rank the documents. Then, the research question is how to exploit the outputs from the base ranker when learning a ranking function from a small number of labeled instances. We refer to this problem as **Ranking Refinement** to distinguish it from supervised learning to rank. Below we show two examples for the application of ranking refinement:

Relevance feedback In information retrieval, documents are often ordered by a predefined relevance ranking function, such as BM25 [51] and Language Model for IR [73], that assesses the relevancy of documents to a given query. Relevance feedback techniques are proposed to improve the retrieval accuracy by allowing users to provide relevance judgments for the first a few retrieved documents. The research question here is how to improve the accuracy of relevance feedback by combining the ranking information from the user feedback as well as the ranking information from the predefined ranking function. We can cast the relevance feedback problem as a rank refinement problem by viewing the relevance ranking function as the base ranker and the documents that are judged by the user as training instances.

Recommender system The goal of a recommender system is to rank the items according to the interest of an active user (i.e., the test user). Usually, a few rated items are provided to indicate the preference of the active user. However, on the other hand, we can rank the items for the active user based on the rating information of the other users using the collaborative filtering techniques [74]. The research question here is how to improve the ranking performance by leveraging the two types of information, i.e. the items rated by the active user and the ranking list generated by the collaborative filtering technique. We cast this problem into the framework of rank refinement by viewing the collaborative filtering algorithm as the base ranker and the rated items as training instances.

Furthermore, any online learning of ranking functions can be viewed as a ranking refinement problem in that the ranking function is updated iteratively with new training examples collected on the fly.

A straightforward approach toward ranking refinement is to view the scores of the base ranker as an additional feature, and learn a ranking function from a limited number of training examples over the augmented features. As will be shown in the experiments, this is not the best approach for exploiting the information hidden in the base ranking function. We believe that the most valuable information behind the base ranker is not its scores but the ranked list of documents it produces. We therefore view the base ranker and the labeled instances as two complementary sources of information, each produces a different loss to evaluate the performance of the new ranking function. The key challenge in combining these two sources of information is that the ranked list generated by the base ranker is imperfect while the labeled instances tend to be noisy. There are two research questions in this problem to address:

Balancing between two sources of relevancy information: The first question is how to balance between two sources of relevancy information; i.e. how to evaluate the effectiveness of a given a ranking function that orders the documents for each query. This question is directly related to the design of the loss function. The common approach in machine learning to balance between two sources of losses is to linearly combine them with a constant. Since the reliability of each source is unknown, finding a good balance parameter is critical in this case. We propose the multiplication of the losses related to two sources of information as an effective and parameter free approach to combine them and show that it satisfies the Pareto Optimality condition [75].

Learning: Given the multiplicative approach for balancing between two different sources (i.e., the base ranker and the training examples), the second research question is how

Notice the application of cross-validation is not possible here since no reliable source of information, i.e. the correct ordering of documents, is available in this case

to learn a ranking function by effectively combining these sources. Our approach to answer this question is based on the boosting framework.

Our empirical study with relevance feedback and recommender system show that the boosting algorithm with multiplicative loss function is effective for ranking refinement, and significantly outperforms the baseline algorithms that incorporate the outputs from the base ranker as an additional feature for the documents.

4.2 Related Work

Most learning to rank algorithms are designed for the setting of supervised learning, in which a ranking function is learned from labeled instances. However, the problem of semi-supervised ranking, the topic of this chapter, has not been addressed in the literature, to the best of our knowledge. The algorithm developed in this chapter belongs to pairwise approach to learning to rank and is closely related to relevance feedback. Therefore, we describe a short bibliography of these two.

Three well-known pairwise approaches to learning to rank are Ranking-SVM [11, 76], RankBoost [19], and RankNet [20]. Ranking-SVM minimizes the number of incorrectly ordered pairs within the maximum margin framework. Several variants [21, 77] are developed to further enhance the performance of Ranking-SVM. RankBoost learns a ranking model based on the same consideration, but by means of Boosting. RankNet [20] is a neural network based approach that uses cross entropy as its loss function.

The relevance feedback techniques [78] are developed to improve the accuracy of the existing retrieval algorithms. There are two types of relevance feedback. The first type, termed user relevance feedback, enhances the retrieval accuracy by collecting the user relevance judgments for the documents that are ranked on the top of the list. As pointed out in the introduction section, the user relevance feedback problem can be treated as a prob-

For the list of different approaches to learning to rank, refer to Chapter 3

lem of ranking refinement. As we showed in the empirical study, the proposed algorithm for ranking refinement significantly outperforms the standard relevance feedback algorithm (i.e., the Rocchio algorithm) over several datasets. The second type of relevance feedback, often termed pseudo relevance feedback, does not explicitly collect the user relevance judgments. Instead, it treats the top ranked documents as relevant to the given query, and the documents ranked at the bottom as irrelevant. These pseudo relevance judgments are used to improve the existing ranking function. It is well known in information retrieval that pseudo relevance feedback may result in degradation of retrieval performance given the high probability of errors in pseudo relevance judgments [78]. This is similar to the noise of training instances in ranking refinement.

4.3 Ranking Refinement

4.3.1 Problem Definition

Let $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ denote the set of instances to be ordered, where each instance $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of d dimensions. Let $G : \mathbb{R}^d \rightarrow \mathbb{R}$ denote the base ranking function (base ranker), and $g_i = G(\mathbf{x}_i)$ denote the ranking score assigned to \mathbf{x}_i by the base ranking function G . Instance \mathbf{x}_i is ranked before \mathbf{x}_j if $g_i > g_j$. To make our problem general, we assume the label information collected from user feedback is presented as a set of ordered pairs, denoted by $\mathcal{O} = \{(\mathbf{x}_{i_k} \succ \mathbf{x}_{j_k}) | k = 1, \dots, m\}$ where each pair $\mathbf{x}_i \succ \mathbf{x}_j$ indicates that instance \mathbf{x}_i is ranked before \mathbf{x}_j . The goal of ranking refinement is to learn a ranking function $F : \mathbb{R}^d \rightarrow \mathbb{R}$ by exploiting both the labeled pairs in \mathcal{O} and the ranking information given by G .

This is because any labeled instances can be converted into ordered pairs while the converse is not true.

4.3.2 Encoding Ranking Information

The first important question for ranking refinement is how to encode the ranking information provided by the base ranking function G . A straightforward approach is to use the ranking scores computed by G as an additional feature, and apply the existing algorithms, such as RankBoost [19] and Ranking-SVM [76], to learn a ranking function from the labeled instances. The drawback of this approach is twofold:

- First, this approach only utilizes the ranking scores of the labeled instances. The ranking information generated by the base ranker for the unlabeled instances is completely ignored by this approach. However, base ranker is a rich source of information for the unlabeled instances that can be exploited for a better ranking. This is particularly important when the number of labeled instances collected from the users's feedback is considerably small.
- Second, we believe that the ranking orders generated by the base ranking function is substantially more reliable than the numerical values of the ranking scores. Similar observation is found in the study of meta search whose goal is to combine the retrieval results of multiple search engines to create a better ranking list [79]. Empirical studies [79] showed that the meta search algorithms based on the document ranks often outperform the algorithms that directly use the relevance scores.

To address the above problems, we encode the order information generated by the base ranking function G with matrix $W \in [0, 1]^{n \times n}$. Each $W_{i,j}$ in the matrix represents the probability of ranking x_i before x_j and is defined as follows

$$W_{i,j} = \frac{\exp(\lambda g_i)}{\exp(\lambda g_i) + \exp(\lambda g_j)} \quad (4.1)$$

In the above, $W_{i,j}$ is defined by a softmax function and the parameter $\lambda \geq 0$ represents the confidence of the base ranking function. To see the effect of λ , we consider two extreme

cases:

- $\lambda = 0$. In this case, we have $W_{i,j} = 0.5$, which indicates that the ordering information generated by the base ranker is completely ignored.
- $\lambda = \infty$. In this case, we have

$$W_{i,j} = \begin{cases} 1 & g_i > g_j \\ 0.5 & g_i = g_j \\ 0 & g_i < g_j \end{cases} \quad (4.2)$$

Thus, W is almost a binary matrix, implying that we completely trust the ranking list generated by the base ranker.

In our experiment, we set λ to be inverse to the standard deviation of the ranking scores for the first 10 retrieved documents.

Similarly, we encode the ordering information inside the set \mathcal{O} with matrix T as follows:

$$T_{i,j} = \begin{cases} 1 - \eta/2 & (\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O} \\ \eta/2 & \text{otherwise} \end{cases} \quad (4.3)$$

where parameter $\eta \in [0, 1]$. $T_{i,j}$ represents the probability of ranking \mathbf{x}_i before \mathbf{x}_j in the training data. The parameter η reflects the error rate of training data, and is particularly useful when the labeled instances are derived from *implicit* user feedback that is usually noisy. In our experiment, we set $\eta = 1/2$.

4.3.3 Objective Function

The goal of ranking refinement is to learn a ranking function $F : \mathbb{R}^d \rightarrow \mathbb{R}$ from matrix W and T that produces a more accurate ranking list than the base ranking function G . In particular, the optimal ranking function F should be consistent with the ranking information

in W and T . To this end, we measure the ranking errors of F with respect to both W and F , i.e.,

$$err_w = \sum_{i,j=1}^n W_{i,j} I(F_j \geq F_i) \quad (4.4)$$

$$err_t = \sum_{i,j=1}^n T_{i,j} I(F_j \geq F_i) \quad (4.5)$$

In the above, we introduce $F_i = F(\mathbf{x}_i)$ and the indicator function $I(x)$ that outputs 1 when the input boolean variable x is true and zero otherwise. There are two problems with directly using the ranking errors err_w and err_t as the objective function:

- First, both error functions are non-smooth functions since the indicator function $I(x)$ is non-smooth. It is well-known that optimizing a non-smooth function is computationally more challenging than optimizing a smooth one [80].
- Second, with two objectives at hand, the problem is essentially a multi-objective optimization problem [75]. Thus, another important question is how to combine multiple objectives into one single objective.

In what follows, we will address these two questions separately.

Relaxation with Exponential Functions. To address the problem with non-smooth objective functions, we follow the idea of boosting by replacing the indicator function $I(x \geq y)$ with an exponential function $\exp(x - y)$. The resulting new objective functions are:

$$\widehat{err}_w = \sum_{i,j=1}^n W_{i,j} \exp(F_j - F_i) \quad (4.6)$$

$$\widehat{err}_t = \sum_{i,j=1}^n T_{i,j} \exp(F_j - F_i) \quad (4.7)$$

Note that since $\exp(x - y) \geq I(x \geq y)$, by minimizing the errors \widehat{err}_w and \widehat{err}_t , we are effective in reducing the original ranking errors err_w and err_t . Another advantage of using \widehat{err}_w and \widehat{err}_t comes from the theoretic result of AdaBoost [81], i.e., by minimizing the exponential loss function, the resulting classifier will not only reduce the training errors but also maximize the classification margin. The enlarged classification margin is the key to guarantee a low generalization error for testing instances [81].

Remark: It is interesting to examine the effect of the smoothing parameter η on the ranking error \widehat{err}_t . By substituting the expression (4.3) for $T_{i,j}$ in (4.7), we have \widehat{err}_t expressed as follows:

$$\begin{aligned}
\widehat{err}_t &= (1 - \eta) \sum_{(\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O}} \exp(F_j - F_i) \\
&\quad + \frac{\eta}{2} \sum_{i,j=1}^n [\exp(F_i - F_j) + \exp(F_j - F_i)] \\
&\approx (1 - \eta) \sum_{(\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O}} \exp(F_j - F_i) + \frac{\eta}{2} \sum_{i,j=1}^n (F_i - F_j)^2 \\
&= (1 - \eta) \left(\sum_{(\mathbf{x}_i \succ \mathbf{x}_j) \in \mathcal{O}} \exp(F_j - F_i) + \frac{\eta}{2(1 - \eta)} \|\mathbf{F}\|_S^2 \right) \tag{4.8}
\end{aligned}$$

where $\|\mathbf{F}\|_S^2$ is a norm of vector $\mathbf{F} = (F_1, \dots, F_n)$ defined as follows:

$$\|\mathbf{F}\|_S^2 = \mathbf{F}^\top (n\mathbf{1} - \mathbf{e}\mathbf{e})\mathbf{F}$$

where $\mathbf{1}$ is the identity matrix and \mathbf{e} is a vector of all ones. In the second step, the approximation follows the Taylor expansion of the exponential function. The second term in (4.8), i.e., $\eta\|\mathbf{F}\|_S^2/2(1 - \eta)$, plays a similar role as used by Support Vector Machines (SVM) [3]. In this sense, the parameter η essentially regularizes the ranking error \widehat{err}_t .

Combining Two Objectives. The problem of optimizing multiple objectives is usually called multi-objective optimization problem [75]. In a multi-objective problem, there is usually no single solution that can satisfy each objective to its fullest. In this problem, we are looking for a solution at which no objective can be further reduced without increasing the value of other objective functions, a condition known as Pareto optimality. The easiest approach to combine several objective functions that results in a Pareto optimal solution is to linearly combine them [75]. In our case, there are two error functions, each related to a different source of relevancy information. A given ranking function can satisfy only one source of relevancy information for each pair of documents in case of a conflict between two sources; i.e. decreasing the error related to one source can increase the error related to another. The linear combination leads to the following optimization problem, i.e.,

$$L_a = \gamma \widehat{err}_w + \widehat{err}_t = \sum_{i,j=1}^n (\gamma W_{i,j} + T_{i,j}) \exp(F_j - F_i) \quad (4.9)$$

where parameter γ is used to weight error \widehat{err}_w . We refer to the approach based on the above objective function as “**Linear Ranking Refinement**”, or **LRR**, for short. The main drawback with the linear combination approach is how to decide the value for γ . In our experiments, we will show that different γ could result in very different performance in information retrieval. Since there is no easy way to find the best tradeoff, we consider the combination of the two errors by their products, i.e.,

$$L_p = \widehat{err}_w \times \widehat{err}_t = \left(\sum_{i,j=1}^n T_{i,j} \exp(F_j - F_i) \right) \left(\sum_{i,j=1}^n W_{i,j} \exp(F_j - F_i) \right) \quad (4.10)$$

We refer to the approach as “**Multiplicative Ranking Refinement**”, or **MRR** for short.

Now, the question is whether the resulting solution is Pareto efficient [75]. More formally, a solution $\mathbf{F} = (F_1, \dots, F_n)$ is Pareto optimal for the objectives \widehat{err}_w and \widehat{err}_t if

there does not exist any other solution $\mathbf{F}' = (F'_1, \dots, F'_n)$ that is either

1. $\widehat{err}_w(F') < \widehat{err}_w(F)$ and $\widehat{err}_t(F') \leq \widehat{err}_t(F)$, or
2. $\widehat{err}_w(F') \leq \widehat{err}_w(F)$ and $\widehat{err}_t(F') < \widehat{err}_t(F)$.

In other words, if \mathbf{F} is Pareto efficient, it guarantees that no solution is able to further reduce the two objectives simultaneously than \mathbf{F} . Regarding the Pareto efficiency when minimizing L_p in (4.10), we have the following theorem:

Theorem 8. *The optimal solution $\mathbf{F} = (F_1, \dots, F_n)$ found by minimizing the objective function L_p is Pareto efficient.*

The proof of this theorem can be found in Appendix A.9. The main advantage of using L_p rather than L_a is that it does not need a weight parameter. This will be revealed in our empirical studies in that minimization of L_p usually significantly outperforms minimization of L_a even when the optimal combination weight γ is used for L_a .

In order to compare the properties of the two different approaches for combination, we examine their first order derivatives. Let ξ denote the parameters used by the ranking function $F(\mathbf{x})$. Then, the first order derivatives of L_a and L_p with respect to ξ are given as follows:

$$\begin{aligned}\nabla_{\xi} L_a &= \sum_{i,j=1}^n (T_{i,j} + \gamma W_{i,j}) \exp(F_j - F_i) (\nabla_{\xi} F(\mathbf{x}_j) - \nabla_{\xi} F(\mathbf{x}_i)) \\ \nabla_{\xi} L_p &= L_p \left(\sum_{i,j=1}^n (a_{i,j} + b_{i,j}) \exp(F_j - F_i) (\nabla_{\xi} F(\mathbf{x}_j) - \nabla_{\xi} F(\mathbf{x}_i)) \right)\end{aligned}$$

where

$$a_{i,j} = \frac{W_{i,j} \exp(F_j - F_i)}{\sum_{i,j=1}^n W_{i,j} \exp(F_j - F_i)} \quad (4.11)$$

$$b_{i,j} = \frac{T_{i,j} \exp(F_j - F_i)}{\sum_{i,j=1}^n T_{i,j} \exp(F_j - F_i)} \quad (4.12)$$

Note that both derivative shares similar structures. The key difference between $\nabla_{\xi} L_a$ and $\nabla_{\xi} L_p$ is that in $\nabla_{\xi} L_p$, $a_{i,j}$ and $b_{i,j}$ are used to weight the contribution from W and T for instance pair $(\mathbf{x}_i, \mathbf{x}_j)$ when computing the derivative. This is in contrast to $\nabla_{\xi} L_a$ where the weights for instance pair $(\mathbf{x}_i, \mathbf{x}_j)$ are $\gamma W_{i,j} \exp(F_j - F_i)$ and $T_{i,j} \exp(F_j - F_i)$. The main advantage of using $a_{i,j}$ and $b_{i,j}$ is that they are normalized, i.e., $\sum_{i,j=1}^n a_{i,j} = \sum_{i,j=1}^n b_{i,j} = 1$, and therefore the contributions from W and T are naturally balanced when calculating the derivative.

4.3.4 Boosting Algorithm for Ranking Refinement

In this section, we will consider algorithms for learning the ranking function $F(\mathbf{x})$ by respectively minimizing the objective function L_p . The objective function L_a is similar to the objective function used by Rank-Boost [19] except that a weight $(T_{i,j} + \gamma W_{i,j})$ is used for each instance pair. We thus can simply modify the Rank-Boost algorithm to learn the optimal ranking function $F(\mathbf{x})$. Hence, in the sequel, we will focus on the boosting algorithm for minimizing L_p .

To learn the optimal ranking function $F(\mathbf{x})$ by minimizing L_p , we follow the greedy approach of boosting algorithms. Since the training examples are the labeled instance pairs, a straightforward boosting approach is to iteratively update the weights of instance pairs and train a new ranking function for the given weighted pairs. This is the strategy employed in the RankBoost algorithm [19]. However, since the number of instance pairs is $\mathcal{O}(n^2)$, this approach could be computationally expensive when the number of instance n is large.

To address the above problem, we present a new boosting algorithm that converts the weights of instance pairs into weights for individual instances. The key idea behind the new boosting algorithm is to derive an upper bound for the target objective that decouples functions for pairs of instances into functions for individual instances. It is this decoupling that makes it possible to infer weights for individual instances from weights for instance

Algorithm 6 Boosting algorithm for minimizing L_p

- 1: **Input:** $W_{i,j}$ and $T_{i,j}$ as two encrypted sources of information
- 2: **repeat**
- 3: Compute $\gamma_{i,j}$ for each instance pair as $\gamma_{i,j} = a_{i,j} + b_{i,j}$ where $a_{i,j}$ and $b_{i,j}$ are defined in (4.11) and (4.12).
- 4: Compute the weight for each instance as $w_i = \sum_{j=1}^n \gamma_{i,j} - \gamma_{j,i}$
- 5: Assign each instance the class label $y_i = \text{sign}(w_i)$.
- 6: Train a classifier $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \{0, 1\}$ that maximizes the following quantity

$$\eta_t = \sum_{i=1}^n |w_i| f(\mathbf{x}_i) y_i \quad (4.13)$$

- 7: Predict f_i for all instances in \mathcal{D}
 - 8: Compute combination weights $\alpha = \frac{1}{2} \log \frac{\sum_{i,j=1}^n \gamma_{i,j} \delta(f_i, 1) \delta(f_j, 0)}{\sum_{i,j=1}^n \gamma_{i,j} \delta(f_j, 1) \delta(f_i, 0)}$ where $f_i = f(\mathbf{x}_i)$. $\delta(x, y)$ outputs 1 if $x = y$ and zero otherwise.
 - 9: Update the ranking function as $F(\mathbf{x}) \leftarrow F(\mathbf{x}) + \alpha f(\mathbf{x})$
 - 10: **until** reach the maximum number of iterations
-

pairs. In addition, the new boosting algorithm is able to derive an appropriate binary class label for each instance using the computed weights. Using both the weights and the class assignments of instances, we can train a binary classifier $f : \mathbb{R}^d \rightarrow \{0, +1\}$ and update the overall ranking function by $F'(\mathbf{x}) = F(\mathbf{x}) + \alpha f(\mathbf{x})$ where α is the combination weight. Note that by converting a ranking problem into a series of binary classification problems, the new boosting algorithm avoids the high computational cost arising from the large number of instance pairs.

Algorithm 6 summarizes the overall procedures for the proposed boosting algorithm minimizing L_p . In each iteration, this algorithm computes $\gamma_{i,j}$ for every pair of instances that measures the uncertainty of ranking instance \mathbf{x}_i ahead of \mathbf{x}_j . Then, it adds up the uncertainties of comparing instance \mathbf{x}_i to all other instances, which results the calculation of the weight for instance \mathbf{x}_i as $w_i = \sum_{j=1}^n \gamma_{i,j} - \gamma_{j,i}$. The bigger the magnitude of w_i , the large the uncertainty of F is in ranking \mathbf{x}_i . So, the algorithm redistributes weights proportional to the uncertainty w at each iteration. It is important to note that w_i can be both positive and negative. In particular, $w_i > 0$ indicates that the algorithm did not succeed

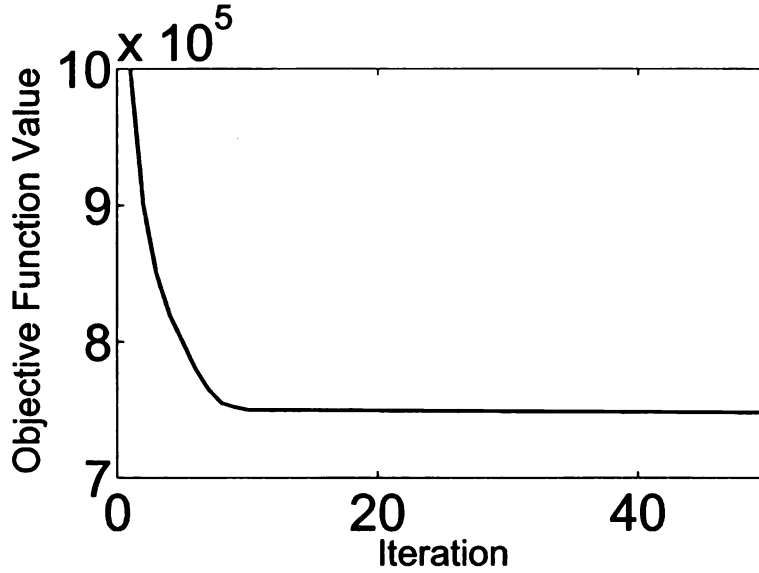


Figure 4.1: Reduction of the objective function L_p using the OHSUMED Data Set

in ranking \mathbf{x}_i on the top of the ranked list; and $w_i \leq 0$ indicates the opposite. Hence, the boosting algorithm derives the class label y_i for \mathbf{x}_i based on the sign of w_i : a positive class for placing instances on the top of the ranked list, and a negative class for placing instances on the bottom of the list. To summarize the above steps, using the magnitude and sign of w_i , the algorithm chooses a weighting and a labeling direction for instances. Given a set of weighted binary class examples, the weak learner trains a classifier that maximizes η_t in (4.13), which can be interpreted as a sort of classification accuracy. Since most binary classifiers are unable to take weights into consideration, the boosting algorithm divides the training procedure into two steps: in the first step, it samples s instances according to the distribution that is proportional to the weights $|w_i|$; It then trains a binary classifier $f : \mathbb{R}^d \rightarrow \{0, +1\}$ using the sampled instances. After learning the new binary classifier in Step 6, the algorithm evaluates its success in reducing the loss (the value of the objective function) in Step 7 and 8 and adds it with a proportional weight to the current list of classifiers at Step 9.

We manually set $s = \max(20, n/5)$ in our empirical study. A similar strategy is employed in the AdaBoost algorithm [19] and its effectiveness has been verified in empirical studies.

Before providing the justification for Algorithm 6, notice that in order to have this algorithm continue iterating, the weak learner need to do better than random guessing in the following sense. Writing α in the following form

$$\alpha = \frac{1}{2} \log \left(\frac{1 - \epsilon}{\epsilon} \right) \quad (4.14)$$

where

$$\epsilon = \frac{\sum_{i,j=1}^n \gamma_{i,j} \delta(f_j, 1) \delta(f_i, 0)}{\sum_{i,j=1}^n \gamma_{i,j} (\delta(f_j, 1) \delta(f_i, 0) + \delta(f_i, 1) \delta(f_j, 0))}$$

applies that by better than random guessing we mean $\epsilon < 0.5$.

In the remaining of this section, we will provide justification to the proposed boosting iterations in Algorithm 6. The main result is summarized in Theorem 9.

Theorem 9. *Let $f^k(\mathbf{x})$ denote the binary classification function obtained in the k th iteration, and $\gamma_{i,j}^k$ denote $\gamma_{i,j}$ learned in that iteration. The objective function after T iterations, denoted by L_p^T , is bounded as follows:*

$$L_p^T \leq \left(\sum_{i,j=1}^n T_{i,j} \sum_{i,j=1}^n W_{i,j} \right) \exp \left(- \sum_{k=1}^T (\sqrt{\mu_k} - \sqrt{\nu_k})^2 \right) \quad (4.15)$$

where

$$\begin{aligned} \mu_k &= \sum_{i,j=1}^n \gamma_{i,j}^k \delta(f^k(\mathbf{x}_i), 1) \delta(f^k(\mathbf{x}_j), 0) \\ \nu_k &= \sum_{i,j=1}^n \gamma_{i,j}^k \delta(f^k(\mathbf{x}_i), 0) \delta(f^k(\mathbf{x}_j), 1) \end{aligned}$$

The above theorem essentially shows that by using the proposed algorithm, the objective function L_p will be reduced exponentially.

The key to proving Theorem 9 is to establish the relationship between the objective function L_p of two consecutive iterations. This is because by upper bounding the log-ratio

between L_p of two consecutive iterations, i.e.,

$$r_t \geq \log L_p^t - \log L_p^{t-1}, \quad (4.16)$$

we will have

$$L_p^T = L_p^0 \prod_{t=1}^T \frac{L_p^t}{L_p^{t-1}} \leq L_p^0 \exp \left(\sum_{t=1}^T r_t \right) \quad (4.17)$$

For the convenience of presentation, in the following, we only consider two consecutive iterations without specifying the index of iteration. Instead, we denote the quantities of the current iteration by symbol $\tilde{\cdot}$ to differentiate the quantities of the previous iteration. In order to establish an upper bound for the log ratio, we first introduce the following lemma

Lemma 5. *Assume $\tilde{F}(\mathbf{x}) = F(\mathbf{x}) + \alpha f(\mathbf{x})$ where $\tilde{F}(\mathbf{x})$ and $F(\mathbf{x})$ are the ranking functions of two consecutive iterations, respectively. $f : \mathbb{R}^d \rightarrow \{0, 1\}$ is a binary classifier and α is the combination weight. We have the following inequality hold for any F , f , and α :*

$$\log \frac{\tilde{L}_p}{L_p} \leq -2 + \sum_{i,j=1}^n (a_{i,j} + b_{i,j}) \exp(\alpha(f_j - f_i)) \quad (4.18)$$

where $a_{i,j}$ and $b_{i,j}$ are defined (4.11) and (4.12), respectively.

The proof of Lemma A.10 can be found in Appendix A.10. Using Lemma A.10, we present the proof of Theorem 9 in Appendix A.11.

Finally, we can show the relationship between the objective function L_p and the quantity η_t (in (4.13)) that is used to guide the training of binary classifiers in iterations. This result is summarized in the following theorem:

Theorem 10. *Let η_k denote the value of the quantity in Equation (4.13) that is maximized by the binary classifier $f^k(\mathbf{x})$ learned in the t th iteration. Assume that $\eta_t \geq 0$ for each iteration. Then, the objective function after T iterations, denoted by L_p^T , is bounded as*

follows:

$$L_p^T \leq \left(\sum_{i,j=1}^n T_{i,j} \right) \left(\sum_{i,j=1}^n W_{i,j} \right) \exp \left(- \sum_{t=1}^T \eta_t \right) \quad (4.19)$$

The proof of the above theorem can be found in Appendix A.12. Theorem 10 provides a theoretical justification for Algorithm 6. In particular, by maximizing η_t , Algorithm 6 effectively reduces the objective function L_p . This is further confirmed by our empirical study. Figure 4.1 shows an example of reduction in the objective function L_p . We clearly see that the objective function is reduced exponentially and receives the largest reduction during the first few iterations.

4.4 Experiments

In this section, we evaluate the proposed algorithm for ranking refinement by two tasks, i.e., user relevance feedback and recommender system. The objectives of our experiments are: (1) to compare the proposed algorithm for ranking refinement to the existing ranking algorithms, (2) to examine the performance of the proposed algorithm for ranking refinement with different numbers of training instances, (3) to examine the effect of different base rankers on the performance of the proposed algorithm, and (4) to examine the time efficiency of the proposed algorithm for ranking refinement. We use the Letor data set for **Relevance Feedback** experiment and Movies data set for the **Recommender System** experiment. The description of these data sets can be found in Section 1.6.2.

4.4.1 Experimental Setup

Algorithms. To examine the effectiveness of the proposed algorithm for ranking refinement, we compared the following ranking algorithms:

Base Ranker: It is the base ranker used in the ranking refinement.

Rocchio: This algorithm extends the standard Rocchio algorithm [82] for user relevance feedback that creates a new query vector by linearly combining the original query vector and vectors of feedback documents. Given the initial query Q_0 , the relevant documents $(R_1, R_2, \dots, R_{n_1})$ and non-relevant documents $(S_1, S_2, \dots, S_{n_2})$, the new query according to Rocchio is computed as:

$$Q = Q_0 + \alpha \sum_{i=1}^{n_1} \frac{R_i}{n_1} - \beta \sum_{i=1}^{n_2} \frac{S_i}{n_2} \quad (4.20)$$

Note, in our case, that each document is not represented by a vector of word frequency, but a vector of features that are computed based on its match to the query. Hence, we don't have Q_0 , i.e., the representation vector for query itself. We therefore set Q_0 to be a vector of all zeros. We used the inner product between the new query and documents as the scores to rank the documents. We vary α and β from 1 to 10 and choose the best setting.

SVM: This implements the Ranking-SVM algorithm using the SVM light package. Note that it is commonly believed that Rank-Boost performs equally well as Ranking SVM. The experimental results provided in the LETOR collection also confirm this. Hence, we only compare the proposal algorithm with Ranking-SVM, but not Rank-Boost.

MRR: This is the Multiplicative Ranking Refinement algorithm that minimizes L_p in (4.10).

LRR: This is the Linear Ranking Refinement algorithm that minimizes L_a in (4.9). Since the performance of LRR depends on the parameter γ , we run LRR with 100 different values from 0.1 to +10 and choose the best and worst performance. We referred them to as **LRR-Worst** and **LRR-Best**, respectively.

For a fair comparison, the output from the base ranker is used as an extra feature when using SVM (i.e., Ranking-SVM) and Rocchio. Notice that we do not compare the performance of the proposed method with different baselines provided in LETOR because the experiments in LETOR are obtained under a different setting. We will discuss the experimental setup used in this chapter in Section 4.4.1. Similar to Chapter 3, we used NDCG to evaluate the performance of different methods. NDCG is described in Section 3.3.1.

Evaluation Protocol. For each LETOR data set, we choose the best ranking feature compared to other features and use it as the base ranker. The best ranker for datasets OHSUMED, TD2003, TD2004, HP2003, HP2004, NP2003, NP2004 are feature number 11, 46, 46, 46, 46, 6, and 6. We followed the common practice of user relevance feedback by collecting the relevance judgments for the first 20 retrieved documents; i.e. we sort all documents of one query based on the base ranker and simulate the user feedbacks by using the true relevancy of the first 20 documents. These user relevance judgments served as labeled instances in ranking refinement. Notice that it is well known that relevance feedback depends on the quality of feedback documents. If the underlying base ranker does a poor job in identifying the relevant documents, it is very likely that most of the feedback documents are irrelevant, leading to a poor performance of the proposed algorithm. We come back to this problem in Section 4.4.3.

For the experiment with recommender system, the base ranker was created by applying a collaborative filtering algorithm, more specifically, the Personality Diagnosis algorithm [74], to the user rating data. In particular, 20 users were randomly selected as the training users, and the remaining 923 users were used for testing. For each test user, 10 rated movies were randomly selected and were used by the collaborative filtering algorithm to identify the 20 training users who share the common interests with the test user. Note that we did not compare the proposed algorithm to other information filtering algorithms because the focus of this study is to examine the effectiveness and the generality of

the proposed approach for ranking refinement.

4.4.2 Results for Relevance Feedback

Figure 4.2 show the ranking results of different algorithms in terms of NDCG for the first 25 ranked documents. First, by comparing the performance of the two variants of ranking refinement, we observed that the Multiplicative Ranking Refinement (MRR) algorithm is more effective than the Linear Ranking Refinement (LLR) algorithm. Indeed, MRR performs significantly better than the best case of LRR (i.e., LRR-best) for OHSUMED and TD2004 datasets. The key difference between MRR and LRR is that MRR minimizes the product of the two error functions while LRR minimizes the weighted sum. We believe it is the normalization scheme brought by MRR (see equations in (4.11) and (4.12)) that makes it performing better than LRR. The performance of MRR is more appreciated given it does not have a single parameter that needs to be adjusted manually.

Second, comparing to the other three baseline algorithms, i.e., the base ranker, Rocchio, Ranking-SVM, we observed that MRR significantly outperforms the base ranker and Rocchio algorithms in all the cases; it outperforms Ranking-SVM in the first three data sets and yields similar performance for the remaining four data sets. We also note that the improvement made by the ranking refinement is more significant for the first a few ranking positions than the other ranking positions, a very desirable property for web search in which users usually only pay attention to the first a few retrieved results. We thus conclude that Multiplicative Ranking Refinement is more effective than the baseline algorithms for user relevance feedback in information retrieval.

Finally, notice that the ranking algorithms show different trend of NDCG on different data sets. Particularly, NDCG is decreasing for the first three data sets and increasing for the remaining data sets. The increasing or decreasing trend is directly dependent on the number of relevant documents and the quality of ranking. If a ranking algorithm performs a good job in retrieving the relevant documents on the top of the list, it is generally expected

to have a decreasing trend. This is because it is more likely to see irrelevant documents in the list as we retrieve more documents. For the last four data sets, there is only one relevant document for each query. Even a good ranker is not able to retrieve the only relevant document on the top of the list and that is why you see NDCG increases until it retrieves the relevant documents of all the queries and then remains constant.

4.4.3 Effect of Base Ranker

We examine how the proposed algorithm response to different base rankers, in particular the base rankers with relatively poor retrieval performance. We tested MRR algorithm with three different base rankers that are selected automatically based on their ranking performance. These three base rankers are the worst, the best and a medium quality base ranker selected from the list of features for each data set. Figure 4.3 shows how MRR algorithm performs when the selected base rankers are used. In each sub-figure, different base rankers are distinguished with a number in the legend that shows the feature number they use. The result indicates that the quality of base rankers has a direct impact on the performance of the MRR algorithm. However, the proposed algorithm is able to significantly improve the performance for a base ranker that can retrieve some relevant documents. When the base ranker performs extremely poor (like in TD2003, HP2003, HP2004, NP2003, and NP2004), all the retrieved documents are are judged as irrelevant by user and no information is available from either sources. Therefore, no improvement can be made by the proposed algorithm for extremely poor base rankers. It is also interesting to observe that for data set OHSUMED, even with the worst base ranker, MMR algorithm is able to achieve similar performance to the baseline methods when they use the best base ranker. This result further confirms the effectiveness of the proposed algorithm for ranking refinement. We thus conclude that the MRR algorithm is resilient to the imperfectness of base rankers.

4.4.4 Effect of Size of Feedback Data

To investigate the effect of the number of feedback documents on the performance, we ran the MRR algorithm by varying the number of feedback documents from 5 to 20. Figure 4.4 shows the result using varied number of feedback documents. We clearly observed that the number of feedback documents have a direct effect on the performance of ranking refinement. However, even with a small amount of feedback, MRR is able to improve the retrieval performance considerably, particularly for the accuracy of the first few ranked documents. We thus conclude that the proposed algorithm for ranking refinement is robust to the size of feedback data. Also notice that for data set NP2003, there is no changes in the performance of MRR with different relevance feedback. The reason is that the base ranker in this case is not able to retrieve any relevant documents for most queries.

4.4.5 Results for Recommender System

We evaluated the generality of the proposed algorithm by applying it to recommender system (movie recommendation). Figure 4.5(a) show the results of different algorithms when applied on the MovieLens dataset. It is surprising to observe that the results of LRR, the linear ranking refinement algorithm, even with the tuned parameter γ , is not comparable to the the performance of the base ranker. In contrast, the MRR algorithm is able to significantly improve the accuracy of the base ranker and outperforms the other baseline algorithms considerably. This result further indicates the importance of appropriately combining the two information sources, i.e., the ranking information behind the base sranker and the feedback information provided by users.

Figure 4.5(b) shows the sensitivity of MRR to the size of feedback data by varying the number of movies rated by the test user from 5 to 25. Similar to the result for relevance feedback, we observed that the size of feedback data affects the performance of MRR considerably. However, even with 5 rated movies, the MRR algorithm is able to make a noticeable improvement in the ranking accuracy compared to the base ranker. This result

further confirms the robustness of the proposed algorithm to the size of feedback data.

4.4.6 Time Efficiency of Ranking Refinement

Figure 4.6 shows the efficiency of the MRR algorithm in terms of the running time for different numbers of rated movies for each test user. We chose movies data set for the experiment because the number of rated movies varies significantly from users to users, making it easy for us to evaluate the computational efficiency of the proposed algorithm. We partitioned the test users into groups where each group of users has a different number of rated movies. The running time of MRR for each group is calculated by averaging it across all the users in the group. As pointed in Section 4.3.4 and seen in Figure 4.6, the running time is linear in the number of instances. Note that the relatively long running time is due to the MATLAB implementation.

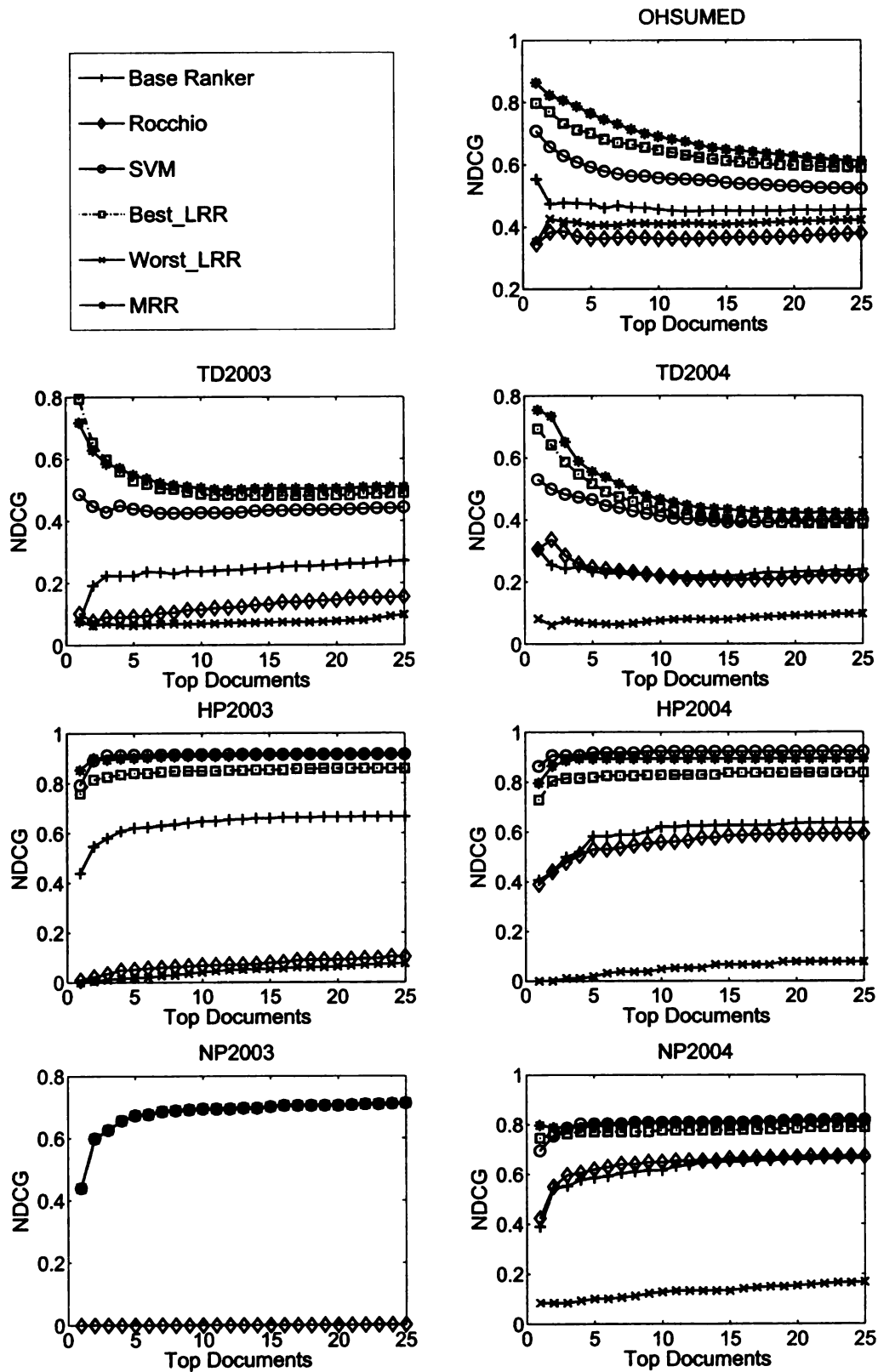


Figure 4.2: NDCG of relevance feedback for different algorithms

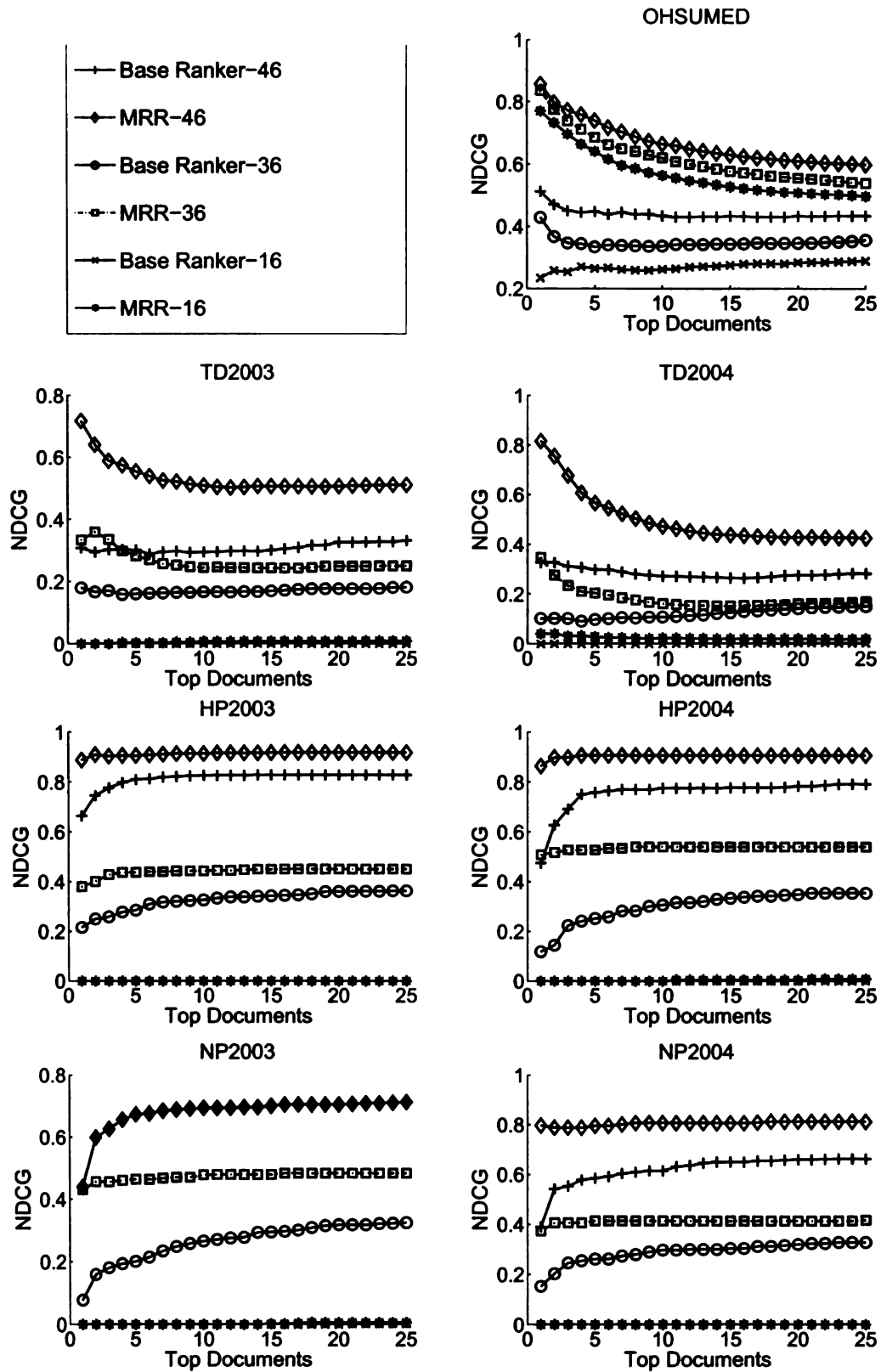


Figure 4.3: NDCG of MRR with different base rankers for relevance feedback

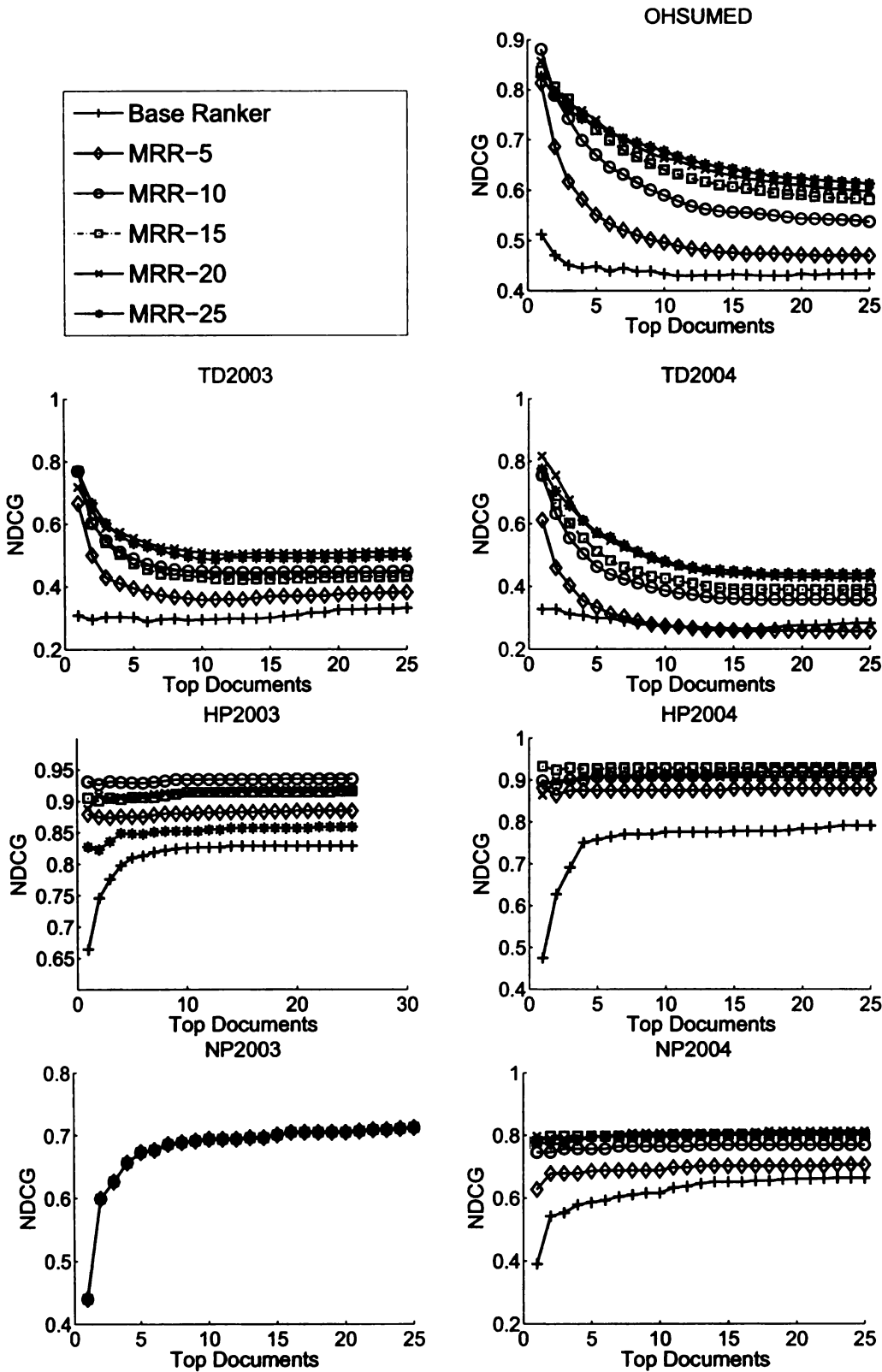


Figure 4.4: NDCG of MRR with different numbers of feedback documents for relevance feedback

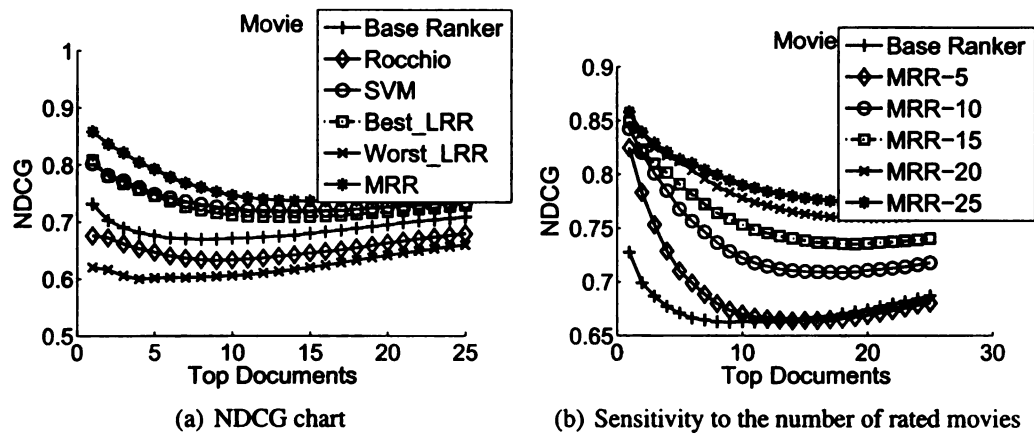


Figure 4.5: The ranking result for recommender system

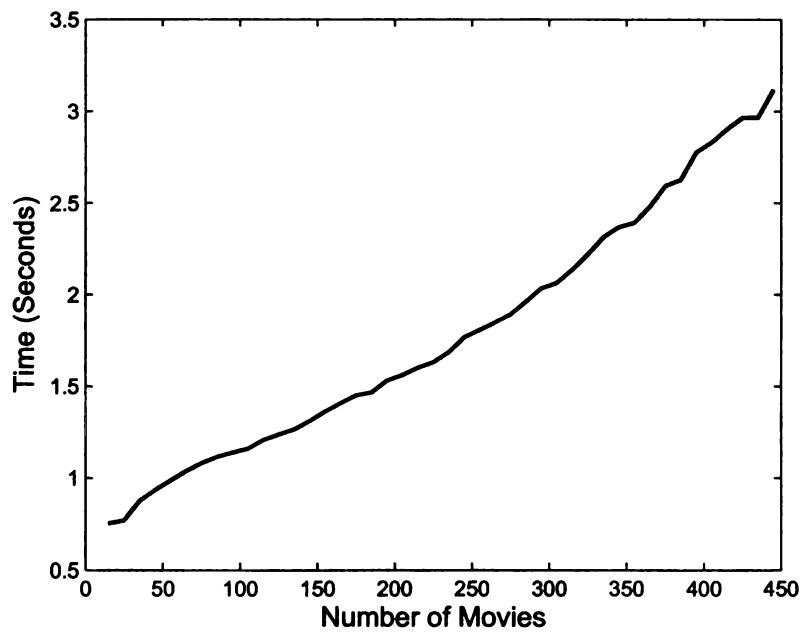


Figure 4.6: Running time of MMR for different numbers of movies rated by test users

Chapter 5

Online Classification with Bandit Feedback

In this chapter, we consider the problem of online classification with bandit feedback: in each trial of online learning, instead of providing the true class label for a given instance, the adversary will only reveal to the learner if the predicted class label is correct. Unlike online learning with full feedback, learner here does not receive the loss value for all the hypotheses in the hypothesis space after it chooses one, which demands a new approach for an effective learning. We present a general framework for online multi-class learning with partial feedback based on the notion of potential [83]. The generality of the proposed framework is verified by the fact that Banditron [5] is indeed its special case with the squared L_2 norm of the weight vector as the potential. Using the exponential potential, we propose an exponential gradient algorithm for online multi-class learning with partial feedback that has the interesting property that its mistake bound is independent from the dimension of data, making it suitable for classifying high dimensional data. Our empirical study with the classification data sets show that the proposed algorithm for online learning with partial feedback is more reliable than Banditron.

5.1 Introduction

Online learning with partial feedback assumes that, in each trial of online learning, the adversary only reveals to the learner if the predicted class label is correct and does not provide the true class label for a given instance. Online learning with partial and full feedback are equivalent when there are only two classes. Therefore, we assume it is clear that the classification problem is a multi-class one when we talk about online classification with bandit feedback.

Online learning with partial feedback is closely related to the problem of multi-armed bandit which is the generalization of a traditional slot machine game, called one armed bandit [84]. In multi-armed bandit, there are n arms to pull with unknown rewards. A player aims to maximize its reward over the trials by learning the best arm to pull. When the player starts, he/she does not know which arm is more profitable. It is only over the trials that he/she learns the best arm to pull. In each stage of this game, the player needs to decide if he/she is going to explore a new arm or exploit his/her knowledge by choosing the best arm, a technique called exploration vs. exploitation tradeoff. This strategy helps the player to constantly receive feedback for all arms.

The problem of online classification with bandit feedback can be considered a multi-armed bandit problem, with the feature vector of example available as a sort of side information; i.e., at each round, after observing an instance, the learner needs to decide a class label (an arm). Although online multi-class learning with full feedback has been extensively studied, the problem of online multi-class learning with partial feedback is only studied recently [5, 85]. The challenge in online learning with bandit feedback is the fact that after classifying a new instance, the learner only receives the loss value for the part of the hypothesis space that have the same prediction as current hypothesis. To explore different parts of the hypothesis space, the learner needs to sacrifice the chance of correctly classifying the current instance in the hope that it finds the best model that minimizes the long-term number of mistakes. We will give a detailed description of this strategy and its

characteristics in Chapter 6.

In this chapter, we propose a general framework to address the challenge of partial feedback in the setup of online classification. This general framework adapts the potential-based gradient descent approaches for online learning [83] to the scenario of partial feedback. The generality of the proposed framework is verified by the fact that banditron is indeed a special case of our framework if the potential function is set to be the squared L_2 norm of the weight vector. Besides the general framework, we further propose an exponential gradient algorithm for online multi-class learning with partial feedback. Compared to the Banditron algorithm, the exponential gradient algorithm is advantageous in that its mistake bound is independent from the dimension of data, making it suitable for classifying high dimensional data. We verify the efficacy of the proposed algorithm for online learning with partial feedback by an extensive empirical study.

5.2 Related Work

Although introduced very recently and there is only a few work directly related, the problem of online multi-class learning with bandit feedback can be traced back to online multi-class classification with full feedback and multi-armed bandit learning. The former provides the required tools to handle the problem of partial feedback and the later offers a starting point for the development of an online multi-class learning with partial feedback. Both these areas have been extensively studied and we only provide a brief review. Several additive and multiplicative online multi-class learning algorithms have been introduced in the literature [52]. Perceptron [43] and Winnow [86] are two such algorithms. Kivinen and Warmuth developed potential functions that can be used to analyze different online algorithms [87]. Grove et al. [88] showed that polynomial potential can be considered as a parameterized interpolation between additive and multiplicative algorithms.

Multi-armed bandit problem refers to the problem of choosing an action from a list

of actions to maximize reward given that the feedback is (bandit) partial [44, 89, 90]. The algorithms developed for this problem usually utilize the exploitation vs. exploitation trade-off strategy to handle the challenge with partial feedback [46, 47].

Multi-class learning with bandit feedback can be considered as a multi-armed bandit problem with side information. Langford et al. in [85] extended the multi-armed setting to the case where some side information is provided. Their setting has a high level of abstraction and its application to the multi-class bandit learning is not straightforward. Banditron, which can be considered as a special case of our framework, is a direct generalization of Perceptron to the case of partial feedback and uses exploration vs. exploitation tradeoff strategy to handle partial feedback [5]. Potential function and exploration vs. exploitation tradeoff techniques are the main tools used to develop the framework in this paper.

Notice that the problem of bandit with side information has been also addressed in reinforcement learning under the name of Associative Bandit problems [91–94]; however those work assume that the side information are i.i.d samples from an unknown distribution. This is unlike our online approach that no assumption is made about the process that generates data.

5.3 A Potential-based Framework for Classification with Partial Feedback

We first present the problem of online classification with partial feedback, followed by the presentation of potential based framework and exponential gradient algorithm.

5.3.1 Problem Definition

We denote by K the number of classes, and by $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$ the sequence of training examples received over trials, where $\mathbf{x}_i \in \mathbb{R}^d$ and T is the number of received training instances. In each trial, we denote by $\tilde{y}_i \in \{1, \dots, K\}$ the predicted class label. Unlike

the classical setup of online learning where an oracle provides the true class label $y_i \in \{1, \dots, K\}$ to the learner, in the case of partial feedback, the oracle only tells the learner if the predicted class label is correct, i.e., $[y_t = \tilde{y}_t]$. This partial feedback makes it difficult to learn a multi-class classification model.

In our study, we assume a linear classifier for each class, denoted by $W = (\mathbf{w}_1, \dots, \mathbf{w}_K) \in \mathbb{R}^{d \times K}$, although the extension to nonlinear classifiers using kernel trick is straightforward. Given a training example (\mathbf{x}, y) , we measure its loss by $\ell \left(\max_{k \neq y} \mathbf{w}_k^\top \mathbf{x} - \mathbf{w}_y^\top \mathbf{x} \right)$ where $\ell(z) = \max(0, z + 1)$ is a hinge loss. We denote by W^1, \dots, W^T a sequence of linear classifiers generated by an online learning algorithm over the trials. Our objective is to bound the number of mistakes made by the online learning algorithm. Since the proposed framework is a stochastic algorithm, we will focus on the expectation of the mistake bound. As will be shown later, the expectation of the mistake bound is often written in the form

$$\alpha \Phi(U) + \beta \sum_{t=1}^T \ell \left(\max_{k \neq y_t} \mathbf{x}_t^\top \mathbf{u}_k - \mathbf{x}_t^\top \mathbf{u}_{y_t} \right)$$

where $U = (\mathbf{u}_1, \dots, \mathbf{u}_K)$ is the linear classifier, $\Phi(W) : \mathbb{R}^{d \times K} \mapsto \mathbb{R}$ is a strictly convex function that measures the complexity of the linear classifiers, and α and β are weight constants for the complexity term and the classification errors. Note that the Banditron algorithm is a special case of the above framework where it measures the complexity of W by its Frobenius norm, i.e., $\Phi(W) = \frac{1}{2} \|W\|_F^2$. In this chapter, we design a general approach for online learning with partial feedback that is adapted to any complexity measure $\Phi(W)$. Finally, for the convenience of presentation, we define

$$\ell_t(W) = \ell \left(\max_{k \neq y_t} \mathbf{x}_t^\top \mathbf{w}_k^t - \mathbf{x}_t^\top \mathbf{w}_{y_t}^t \right) \quad (5.1)$$

5.3.2 Banditron

Kakade et al. [5] developed Banditron for the problem of online classification with bandit feedback. Banditron, depicted in Algorithm 7, is basically Perceptron adapted to handle the case of bandit feedback by utilizing the exploration vs. exploitation tradeoff technique. After receiving a new instance x_t , Banditron computes the primary class assignment \hat{y}_t using the weight matrix W^{t-1} at Step 5, just like Perceptron. Using the exploration vs. exploitation tradeoff parameter γ , the learner decides label \tilde{y}_t at Step 6 and 7 which is either \hat{y}_t (exploitation) or another random class label (exploration). After receiving a feedback, the algorithm computes the update matrix $x_t \delta_t$ which, on average, is equivalent to the update matrix in Perceptron for the full feedback setting. Kakade et al. provided the following mistake bound for Banditron in [5].

Bound for Banditron: Let K be the number of classes. After running over a sequence of examples $\mathbf{x}_1, \dots, \mathbf{x}_T$, with $\|\mathbf{x}_t\|_2 \leq 1$ for all t , the expected number of mistakes made by Banditron, denoted by $E[M]$, is bounded as follows

$$E[M] \leq \ell(U) + \gamma T + 3 \max \left\{ \frac{2K|U|_F^2}{\gamma}, \sqrt{|U|_F^2 \gamma T} \right\} + \sqrt{\frac{2K|U|_F^2 \ell(U)}{\gamma}} \quad (5.2)$$

where U is any arbitrary weight matrix (classifier) and L .

5.3.3 Potential-based Online Classification for Partial Feedback

Our framework, depicted in Algorithm 8, generalizes the Banditron algorithm [5] by considering any complexity measure $\Phi(W)$ that is strictly convex. In this algorithm, we introduce $\theta \in \mathbb{R}^{d \times K}$, the dual representation of the linear classifiers W . In each iteration, we first update θ_t based on the partial feedback $[y_t = \tilde{y}_t]$, and compute the linear classifier W_t via the mapping $\nabla \Phi^*(\theta)$, where $\Phi^*(\theta)$ is the Lagendre conjugate of $\Phi(W)$. Similar to Banditron and most online learning with partial feedback [83], a stochastic approach is used

Algorithm 7 The Banditron Algorithm

- 1: Parameters:
 - Step size: $\gamma > 0$
 - 2: Set $\mathbf{w}_k^0 = \mathbf{0}, k = 1, \dots, K$ and $\theta^0 = \nabla \Phi^*(W^0)$
 - 3: **for** $t = 1, \dots, T$ **do**
 - 4: Receive $\mathbf{x}_t \in \mathbb{R}^d$
 - 5: Compute $\hat{y}_t = \arg \max_{1 \leq k \leq K} \mathbf{x}_t^\top \mathbf{w}_k^{t-1}$
 - 6: Set $p_k = (1 - \gamma)[k = \hat{y}_t] + \gamma/K, k = 1, \dots, K$
 - 7: Sample \tilde{y}_t by distribution $\mathbf{p} = (p_1, \dots, p_K)$.
 - 8: Predict \tilde{y}_t and receive feedback $[y_t = \tilde{y}_t]$
 - 9: Compute $\delta_t = \mathbf{1}_{\hat{y}_t} - \mathbf{1}_{\tilde{y}_t} \frac{[y_t = \tilde{y}_t]}{p_{\tilde{y}_t}}$ where $\mathbf{1}_k$ stands for the vector with all its elements being zero except its k th element is 1.
 - 10: Compute $W^t = W^{t-1} - \mathbf{x}_t \delta_t^\top$
 - 11: **end for**
-

to predict class assignment, in which parameter $\gamma > 0$ is introduced to ensure sufficient exploration [44].

In the following, we show the mistake bound for the proposed algorithm. For the convenience of discussion, we define vector $\tau_t \in \mathbb{R}^d$ as

$$\tau_t = \mathbf{1}_{\hat{y}_t} - \mathbf{1}_{y_t} \quad (5.5)$$

Proposition 3. For $\ell_t(W) \geq 1$, we have

$$\nabla \ell_t(W) = \left\langle W^{t-1}, \mathbf{x}_t \tau_t^\top \right\rangle, \quad \text{and} \quad \mathbb{E}_t[\delta_t] = \tau_t \quad (5.6)$$

where $\mathbb{E}_t[\cdot]$ is the expectation over \tilde{y}_t and δ_t is defined in (5.4).

We denote by $D_\Phi(A, B)$ the Bregman distance function for a given convex function Φ , which is defined as follows

$$D_\Phi(A, B) = \Phi(A) - \Phi(B) - \langle A - B, \nabla \Phi(B) \rangle \quad (5.7)$$

The following classical result in convex analysis summarizes useful properties of Bregman

Algorithm 8 Online Learning Algorithm for Multi-class Bandit Problem

- 1: Parameters:
 - Smoothing parameter: $\gamma \in (0, 0.5)$
 - Step size: $\eta > 0$
 - Potential function: $\Phi : \mathbb{R}^{d \times K} \mapsto \mathbb{R}$ and its Legendre conjugate $\Phi^* : \mathbb{R}^{d \times K} \mapsto \mathbb{R}$
- 2: Set $\mathbf{w}_k^0 = \mathbf{0}, k = 1, \dots, K$ and $\theta^0 = \nabla \Phi^*(W^0)$
- 3: **for** $t = 1, \dots, T$ **do**
- 4: Receive $\mathbf{x}_t \in \mathbb{R}^d$
- 5: Compute

$$\hat{y}_t = \arg \max_{1 \leq k \leq K} \mathbf{x}_t^\top \mathbf{w}_k^{t-1} \quad (5.3)$$

- 6: Set $p_k = (1 - \gamma)[k = \hat{y}_t] + \gamma/K, k = 1, \dots, K$
- 7: Randomly sample \tilde{y}_t according to the distribution $\mathbf{p} = (p_1, \dots, p_K)$.
- 8: Predict \tilde{y}_t and receive feedback $[y_t = \tilde{y}_t]$
- 9: Compute

$$\delta_t = \mathbf{1}_{\hat{y}_t} - \mathbf{1}_{\tilde{y}_t} \frac{[y_t = \tilde{y}_t]}{p_{\hat{y}_t}} \quad (5.4)$$

where $\mathbf{1}_k$ stands for the vector with all its elements being zero except its k th element is 1.

- 10: Compute $\theta^t = \theta^{t-1} - \eta \mathbf{x}_t \delta_t^\top$
 - 11: Compute $W^t = \nabla \Phi(\theta^t)$ where $\theta^t = (\theta_1^t, \dots, \theta_K^t)$
 - 12: **end for**
-

distance.

Lemma 6. Let $\Phi(W)$ be a strictly convex function with constant ρ with respect to norm $\|\cdot\|$, i.e., for any W and W' we have

$$\langle W - W', \nabla \Phi(W) - \nabla \Phi(W') \rangle \geq \rho \|W - W'\|^2.$$

We have the following inequality for any θ and θ'

$$\langle \theta - \theta', \nabla \Phi^*(\theta) - \nabla \Phi^*(\theta') \rangle \leq \frac{1}{\rho} \|\theta - \theta'\|_*^2$$

where $\Phi^*(\theta)$ is the Legendre conjugate of $\Phi(W)$, and $\|\cdot\|_*$ is dual of norm $\|\cdot\|$. Further-

more, we have the following equality for any W and W'

$$D_{\Phi}(W, W') = D_{\Phi^*}(\theta, \theta'),$$

where $\theta = \nabla \Phi(W)$ and $\theta' = \nabla \Phi(W')$.

Proposition 4. For any linear classifier $U \in \mathbb{R}^{d \times K}$, We have the following inequality hold for two consecutive classifier W^{t-1} and W^t generated by Algorithm 8

$$\begin{aligned} D_{\Phi^*}(U, W^{t-1}) - D_{\Phi^*}(U, W^t) + D_{\Phi^*}(W^{t-1}, W^t) \\ = -\langle U - W^{t-1}, \eta \mathbf{x}_t \delta_t^\top \rangle \end{aligned} \quad (5.8)$$

Proof. Using the property of Bregman distance function (see for example Chapter 11.2 in [83]), we have

$$\begin{aligned} D_{\Phi^*}(U, W^{t-1}) - D_{\Phi^*}(U, W^t) + D_{\Phi^*}(W^{t-1}, W^t) &= \langle U - W^{t-1}, \nabla \Phi^*(W^t) - \nabla \Phi^*(W^{t-1}) \rangle \\ &= \langle U - W^{t-1}, \theta^t - \theta^{t-1} \rangle \\ &= -\langle U - W^{t-1}, \eta \mathbf{x}_t \delta_t^\top \rangle \end{aligned}$$

The second step follows the property $\theta_t = \nabla \Phi^*(W^t)$, and the last step uses the updating rule of Algorithm 8. \square

Now, we can bound $\mathbb{E}[|\delta_t|_s^2]$ as follows, with the proof provided in Appendix A.13.

Proposition 5. For any $s > 0$, we have

$$\mathbb{E}[|\delta_t|_s^2] \leq \frac{\gamma}{1-\gamma} + [\hat{y}_t \neq y_t] \left\{ 1 - \gamma + \frac{\gamma}{K} \left(1 + \left\lceil \frac{K}{\gamma} \right\rceil^s \right)^{2/s} \right\}$$

We use $|W|_{p,s}$ to measure the norm of matrix $W \in \mathbb{R}^{d \times K}$ with $p \geq 1$ and $s \geq 1$. It is

defined as

$$|W|_{p,s} = \max_{|\mathbf{u}|_p \leq 1, |\mathbf{v}|_s \leq 1} \langle \mathbf{u}, W\mathbf{v} \rangle \quad (5.9)$$

where $\mathbf{u} \in \mathbb{R}^d$, $\mathbf{v} \in \mathbb{R}^K$, and $|\mathbf{u}|_q$ and $|\mathbf{v}|_t$ are L_q and L_t norm of vector \mathbf{u} and \mathbf{v} , respectively. Evidently, the dual norm of $|\cdot|_{p,s}$ is $|\cdot|_{q,t}$, with $p^{-1} + q^{-1} = 1$ and $s^{-1} + t^{-1} = 1$. The theorem below shows the regret bound for Algorithm 8. The proof of this theorem is provided in Appendix A.14

Theorem 11. *Assume that for the sequence of examples, $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$, we have, for all t , $\mathbf{x}_t \in \mathbb{R}^d$, $\|\mathbf{x}_t\|_p \leq 1$ and the number of classes is K . Let $U = (\mathbf{u}_1, \dots, \mathbf{u}_K) \in \mathbb{R}^{d \times K}$ be any matrix, and $\Phi^* : \mathbb{R}^{d \times K} \mapsto \mathbb{R}$ be a strictly convex function with constant ρ with respect to norm $|\cdot|_{p,s}$. The expectation of the number of mistakes made by Algorithm 8, denoted by $E[M]$, is bounded as follows*

$$E[M] \leq \frac{1}{\eta\kappa} D_{\Phi^*}(U) + \frac{1}{\kappa} \sum_{t=1}^T \ell_t(U) + \frac{\eta\gamma T}{2\rho\kappa(1-\gamma)} + \gamma T$$

where

$$\kappa = 1 - \frac{\eta}{2\rho} \left\{ 1 - \gamma + \frac{\gamma}{K} \left(1 + \left[\frac{K}{\gamma} \right]^s \right)^{2/s} \right\}$$

Notice that the Banditron algorithm is a special case of the general framework with $\Phi^*(W) = \frac{1}{2} |W|_F^2$ and $|\cdot|_{p,s} = |\cdot|_{2,2} = |\cdot|_F$. The Banditron bound is specifically obtained through approximations $\gamma/(1-\gamma) \leq 2\gamma$ and $1 + k/\gamma \leq 2k/\gamma$ in summarizing the terms in κ .

5.3.4 Exponential Gradient for Online Classification with Partial Feedback

In this section, we extend the exponent gradient algorithm to online multi-class learning with partial feedback. A straightforward approach is to use the result in Theorem 11 by setting

$$\Phi(\theta) = \sum_{k=1}^K \sum_{i=1}^d \exp(\theta_{i,k}) \quad (5.10)$$

$$\Phi^*(W) = \sum_{k=1}^K \sum_{i=1}^d W_{i,k} (\ln W_{i,k} - 1) \quad (5.11)$$

where each \mathbf{w}_k is a probability distribution. Following the general framework presented in Algorithm 8, Algorithm 9 summarizes the exponential gradient algorithm for online multi-class learning with partial feedback. Since $\Phi^*(W)$ is strictly convex with constant 1 with respect to $\|\cdot\|_F$, we have following mistake bound for the exponential gradient algorithm.

Theorem 12. *Assume that for the sequence of examples, $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$, we have, for all t , $\mathbf{x}_t \in \mathbb{R}^d$, $\|\mathbf{x}_t\|_2 \leq 1$ and the number of classes is K . Let $U = (\mathbf{u}_1, \dots, \mathbf{u}_K) \in \mathbb{R}^{d \times K}$ where each \mathbf{u}_k is a distribution. The expectation of the number of mistakes made by Algorithm 9 is bounded as follows*

$$\mathbb{E}[M] \leq \frac{K \ln K}{\kappa \eta} + \frac{1}{\kappa} \sum_{t=1}^T \ell_t(U) + \frac{\eta \gamma T}{2\rho \kappa (1 - \gamma)} + \gamma T$$

where $\kappa = 1 - \frac{\eta}{2\rho} \left(1 - \gamma + \frac{\gamma}{K} + \frac{K}{\gamma}\right)$.

By minimizing the mistake bound in the above theorem, we choose step size η as follows

$$\eta = \sqrt{\frac{K(1 - \gamma) \ln K}{T\gamma}} \quad (5.12)$$

Algorithm 9 Exponential Gradient Algorithm for Online Multi-class Learning with Partial Feedback

- 1: Parameters:
 - Smoothing parameter: $\gamma \in (0, 0.5)$
 - Step size: $\eta > 0$
- 2: Set $\theta^0 = \mathbf{1}\mathbf{1}^\top/d$
- 3: **for** $t = 1, \dots, T$ **do**
- 4: Compute $W_{i,k}^t = \exp(\theta_{i,k}^t)/Z_k^t$ where $Z_k^t = \sum_{i=1}^d \exp(\theta_{i,k}^t)$.
- 5: Receive $\mathbf{x}_t \in \mathbb{R}^d$
- 6: Compute

$$\hat{y}_t = \arg \max_{1 \leq k \leq K} \mathbf{x}_t^\top \mathbf{w}_k^{t-1} \quad (5.13)$$

- 7: Set $p_k = (1 - \gamma)[k = \hat{y}_t] + \gamma/K, k = 1, \dots, K$
- 8: Randomly sample \tilde{y}_t according to the distribution $\mathbf{p} = (p_1, \dots, p_K)$.
- 9: Predict \tilde{y}_t and receive feedback $[y_t = \tilde{y}_t]$
- 10: Compute

$$\delta_t = \mathbf{1}_{\hat{y}_t} - \mathbf{1}_{\tilde{y}_t} \frac{[y_t = \tilde{y}_t]}{p_{\tilde{y}_t}} \quad (5.14)$$

where $\mathbf{1}_k$ stands for the vector of all elements being zero except that its k th element is 1.

- 11: Compute $\theta^t = \theta^{t-1} - \eta \mathbf{x}_t \delta_t^\top$
 - 12: **end for**
-

For the high dimensional data, we can improve the result in Theorem 12 by using the following lemma. The proof of this lemma is provided in A.15.

Lemma 7. $\Phi(W)$ and $\Phi^*(W)$ defined in (5.10) and (5.11) satisfies the following properties

$$\begin{aligned} \langle W - W', \nabla \Phi^*(W) - \nabla \Phi^*(W') \rangle &\geq \sum_{k=1}^K |\mathbf{w}_k - \mathbf{w}'_k|_1^2 \\ \langle \theta - \theta', \nabla \Phi(\theta) - \nabla \Phi(\theta') \rangle &\leq \sum_{k=1}^K |\theta_{*,k} - \theta'_{*,k}|_\infty^2 \end{aligned}$$

where $\theta_{*,k} = (\theta_{1,k}, \dots, \theta_{d,k})$.

Using the above lemma, we have the following theorem that updates the result in Theorem 12

Theorem 13. *Same as the setup of Theorem 12 except that $|\mathbf{x}_i|_\infty \leq 1$. The expectation of the number of mistakes made by Algorithm 9 is bounded as follows*

$$\mathbb{E}[M] \leq \frac{K \ln K}{\kappa \eta} + \frac{1}{\kappa} \sum_{t=1}^T \ell_t(U) + \frac{\eta \gamma T}{2\rho \kappa(1-\gamma)} + \gamma T$$

where $\kappa = 1 - \frac{\eta}{2\rho} \left(2 - 2\gamma - \frac{4\gamma}{K}\right)$.

Proof. The proof is the same as the proof of Theorem 11 except that we have

$$\mathbb{E}[D_\Phi(\theta^{t-1}, \theta^t)] \leq \frac{\eta^2}{2\rho} \mathbb{E} \left[\sum_{k=1}^K |\delta_{t,k}| |\mathbf{x}_t|_\infty^2 \right] \leq \frac{\eta^2}{2\rho} \mathbb{E}[|\delta_t|_1]$$

A simple computation shows that $\mathbb{E}[|\delta_t|_1] = 2 - 2\gamma - 4\gamma/K$. By combining these results, we have the theorem. \square

The major difference between Theorem 12 and 13 is the constraint on \mathbf{x} : L_2 is used in Theorem 12 and L_∞ is used in Theorem 13. Therefore, Theorem 13 shows that the exponential gradient algorithm is essentially independent from dimensionality d , making it suitable for handling high dimensional data.

5.4 Experiments

To study the performance of the proposed framework, we applied the exponential potential algorithm introduced in 5.3.4 on the multi-class classification data sets introduced in Section 1.6.1.

We compared the classification performance of the proposed exponential gradient algorithm, Exp, to the Banditron algorithm. Since the exponential gradient algorithm assumes all the combination weights to be non-negative, in order to make fair comparison between the proposed approach and the Banditron algorithm, we run two sets of experiments for Banditron, one which is the original Banditron and one that projects the learned weights

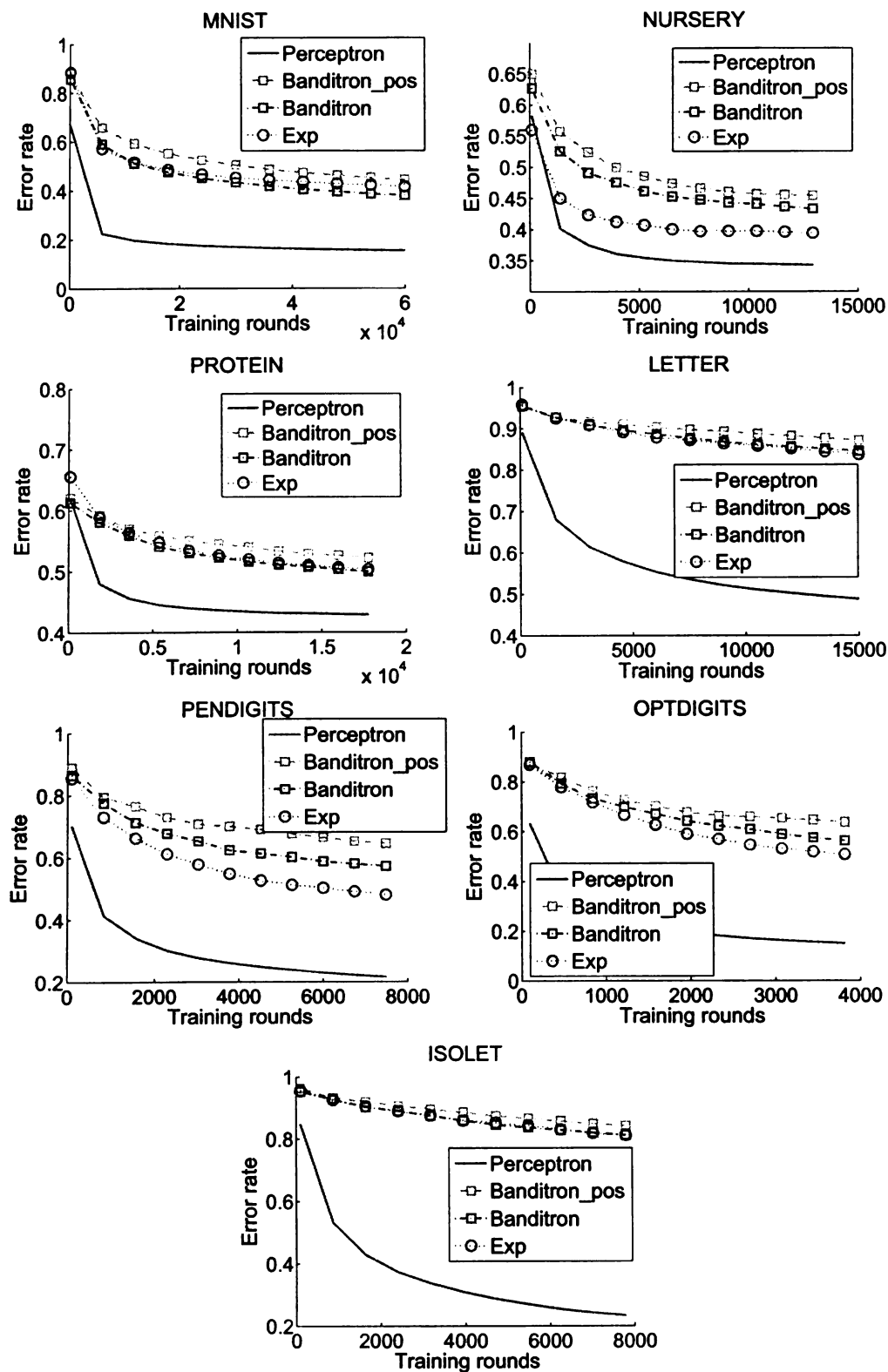


Figure 5.1: The figure shows the error rates of different methods over trials with the best setting of γ .

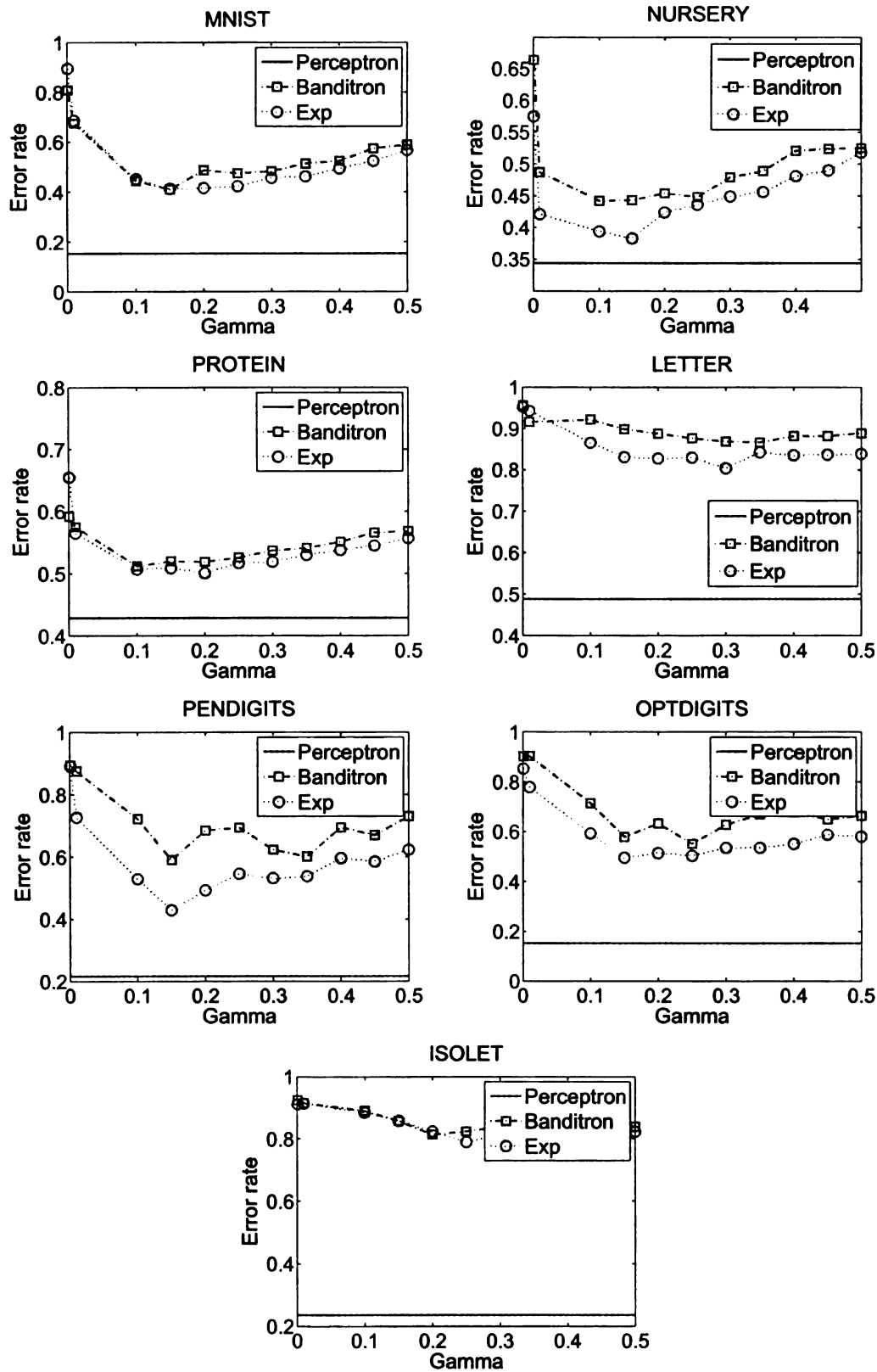


Figure 5.2: Figure shows the final error rates of different methods with varied γ .

into the positive orthants, which is equivalent to setting all the negative weights to be zero. It is easy to verify that the projection step does not change the theoretic properties of Banditron, in particular the mistake bound (of course only with respect to linear classifiers U in the positive orthants). We call this projected Banditron, Banditron_Pos.

For each tested algorithm and for each data set, we conduct 10 independent runs with different seeds for randomization. We evaluate the performance of online learning by the accumulate error rate, which is computed as the ratio of the number of misclassified samples and the number of samples received so far during the online learning process.

Since these online algorithms rely on the parameter γ to control the tradeoff between exploration and exploitation, we examine the classification results of all the algorithms in comparison by varying γ . The step size η of online learning often play an important role in the final performance. For the proposed algorithm, we set the step size according to Eq. 5.12. Because the exponential function may exceed the upper bound of a real number with double precision type in a 64-bit computer, we further multiple the step size with a small factor (typically 10^{-5}) to avoid this issue.

5.4.1 Experimental results

Figure 5.2 compares the average error rates of the online algorithms with varied γ values, and Figure 5.1 shows the average error rates of the three online methods over the entire online process. For the proposed algorithm and both version of Banditron, we choose the optimal γ that results the lowest classification error rate.

First, by examining the classification performance with varied γ , we clearly see that the exponential gradient algorithm shows comparable performance compared with the original Banditron algorithm for online multi-class learning with limited feedback. In particular, we observe that the proposed algorithm performs significantly better than the Banditron algorithm for three data sets 'OptDigits', 'Pendigits', and 'Nursery'. The result indicates that the proposed algorithm is overall more reliable. Notice that for all data sets except for

'Nursery' data set, we observe a significant gap between online learning with full feedback and online learning with partial feedback, which is due to the limited feedback from the adversary.

Second, we compare the learning rate of all three algorithms. We observe that the proposed algorithm overall exhibits a significantly better learning rate than the Banditron_Pos algorithm (i.e. Banditron with positive weights), for most data sets and most part of the online learning process. This result indicates that the proposed online learning algorithm with partial feedback is generally effective in reducing the error rate.

Finally, notice that these algorithms are sensitive to the choice of parameter γ . In Chapter 6, we provide more details on the exploration vs. exploitation tradeoff parameter γ and provide effective algorithm to automatically tune it.

Chapter 6

Robust Online Classification With Bandit Feedback

As we have already seen in Chapter 5, exploration vs. exploitation tradeoff strategy is the main tool to develop online classification algorithms with bandit feedback. The major problem with utilizing this strategy is the sensitivity of the resulting algorithm to the exploration vs. exploitation tradeoff parameter. In this chapter, we propose three learning strategies to automatically adjust the tradeoff parameter for Banidtron. Our extensive empirical study with multiple real-world data sets verifies the efficacy of the proposed approach in learning the exploration vs. exploitation tradeoff parameter.

6.1 Introduction

Exploitation vs. exploration tradeoff strategy has been widely applied to develop online learning techniques when the feedback provided to learner is bandit, i.e. the learner only receives the cost of its action but not the cost of other possible actions. Exploration refers to the choice of an action not recommended as the best action by the current model (classifier). It allows the learner to explore the game and receive the feedback for different strategies and gain new knowledge from the adversary. Exploitation refers to choice of the best action

according to the current knowledge in order to maximize the gain. These two objectives are complementary, but opposite: exploration leads to maximization of the gain in the long run at the risk of losing short term reward; exploitation maximizes the short term gain at the price of losing the gain over the long run. A careful tradeoff between these two objectives is important to the success of any online learner utilizing the combined strategy.

The challenge of online classification with bandit feedback is that after classifying an instance, the learner only receives the loss value for those hypotheses that have the same prediction as the current hypothesis. This means that the learner is not able to explore the whole hypothesis space if it only classifies according to the current hypothesis. As described in Chapter 5, Banditron [5] utilizes the exploration vs. exploitation tradeoff techniques to handle this challenge. This tradeoff is explicitly captured by a single parameter $\gamma \in (0, 0.5)$ in Banditron: with probability $1 - \gamma$, the learner will predict the most likely class label based on the current classification model (exploitation), and with probability γ , the learner will randomly choose one of the remaining class labels for prediction (exploration).

Figure 6.1 shows the performance of Banditron for different data sets by varying the value of γ . The best γ values for data sets 'Protein', 'Pendigits', 'Isolet', 'Nursery', 'Optdigits', 'Letter', and 'Mnist' are respectively 0.1, 0.25, 0.2, 0.15, 0.25, 0.35, and 0.15. It is clear that the performance of Banditron strongly depends on the value of γ and it is therefore very helpful to develop strategies to automatically tune this parameter. Intuitively, at the beginning of the learning stage, due to the fact that classification model is trained by a limited number of examples, it is likely that the classification model will perform poorly. As a result, it may be more desirable to have a large value for γ . As the learning procedure proceeds, the classification model is updated with sufficiently large number of examples, and therefore is likely to yield accurate classification performance. Therefore, it is desirable to reduce the value of γ and the amount of exploration with increasing number of

Notice these plots are extracted from Figure 5.2.

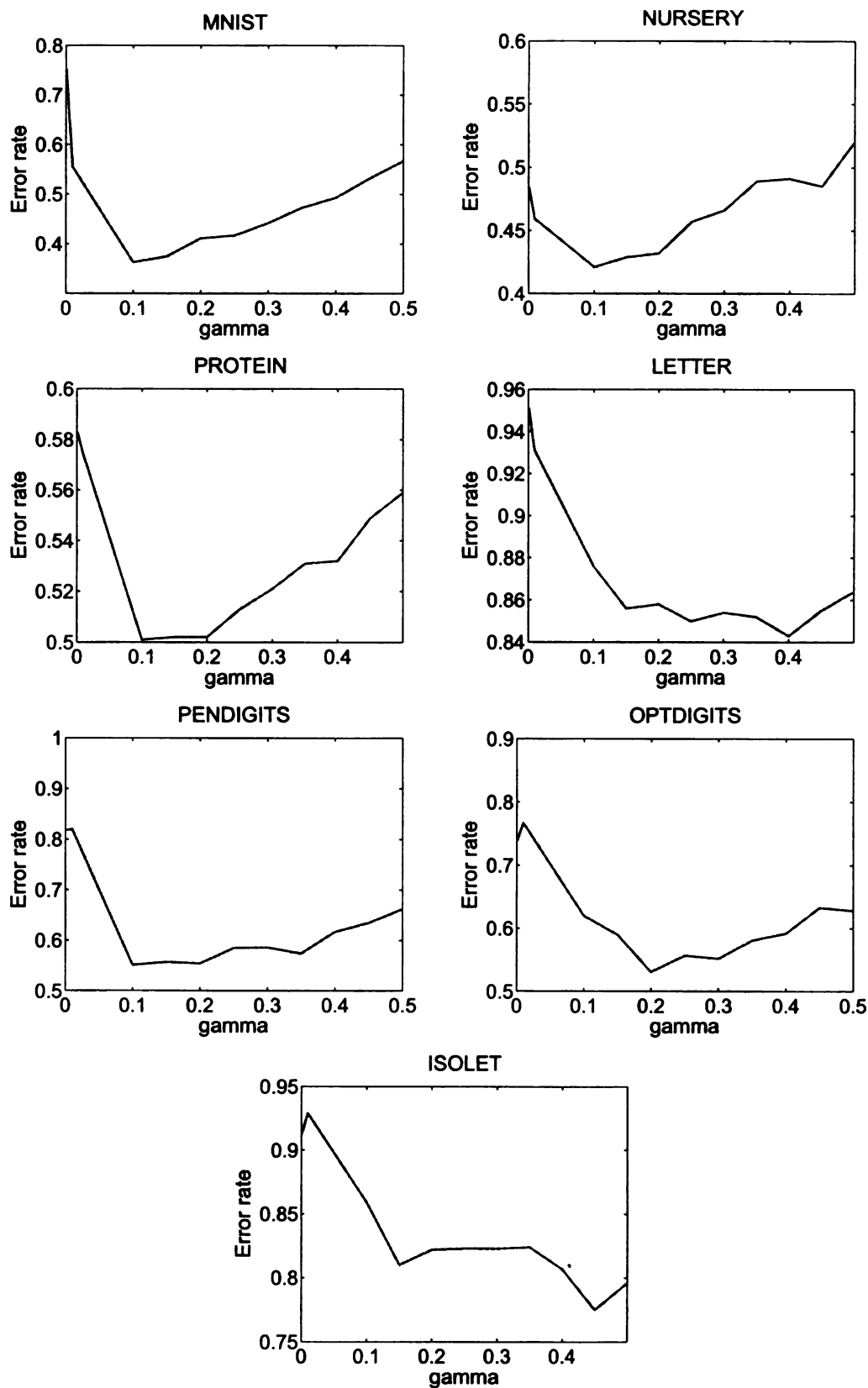


Figure 6.1: The error rates of Banditron with different choice of γ for different data sets

training examples. Theoretically, as suggested by [5], the choice of $\gamma = O(t^{-1/3})$ produces the optimum result in the agnostic case. This is because by minimizing the mistake bound provided in Inequality 5.2 with regard to γ , we obtain $\gamma = O(t^{-1/3})$. However, as we show in this chapter, the adaptive choice of parameter γ should be not only dependent on time t but also dependent on the number of correctly/incorrectly classified instances to control the speed in which we reduce the amount of exploration.

6.2 Related Work

To the best of our knowledge, this is the first study that aims to learn the exploration vs exploitation tradeoff parameter for online classification with bandit feedback. Since exploration vs. exploitation tradeoff parameter is widely utilized for the problem of multi-armed bandit, here we briefly describe the tuning techniques for this parameter in multi-armed bandit [6]. However, none of these methods utilizes the classification specific information, e.g. the number of mistakes.

In the simplest form, called γ -first strategy [6], a pure exploration phase is followed by a pure exploitation phase [46, 95]. Evan-Dar et al. [46] showed that to obtain an α -optimal arm with probability $1 - \delta$, $O(\frac{K}{\alpha^2} \log(\frac{K}{\delta}))$ rounds of exploration is needed. The problem with this approach is that it cannot produce arbitrary small regrets. A second approach, called γ -decreasing strategy [6], is similar to the approach proposed by Kakade et al. [5] for the problem of online classification with bandit feedback. In this approach, γ is a decreasing function of time t . Several decreasing function have been proposed including $\gamma_t = O(\frac{1}{t})$ [6], $\gamma_t = O(\frac{\log(t)}{t})$ [96], and $\gamma_t = O(\frac{1}{t^3})$ [5].

Another approach, the Boltzmann exploration, chooses each arm with a probability proportional of their obtained reward [96]. A temperature parameter can be utilized to smoothly switch from pure exploration to a pure exploitation. Notice that except γ -decreasing strategy proposed in [5], no theoretical results are known for the other methods

Algorithm 10 The Banditron Algorithm

```
1: Set  $\mathbf{w}_k^0 = \mathbf{0}, k = 1, \dots, K$  and  $\theta^0 = \nabla \Phi^*(W^0)$ 
2: for  $t = 1, \dots, T$  do
3:   Receive  $\mathbf{x}_t \in \mathbb{R}^d$ 
4:   Compute  $\hat{y}_t = \arg \max_{1 \leq k \leq K} \mathbf{x}_t^\top \mathbf{w}_k^{t-1}$ 
5:   Choose sampling probability  $\gamma_t$ 
6:   Set  $p_k = (1 - \gamma_t)[k = \hat{y}_t] + \gamma_t/K, k = 1, \dots, K$ 
7:   Sample  $\tilde{y}_t$  by distribution  $\mathbf{p} = (p_1, \dots, p_K)$ .
8:   Predict  $\tilde{y}_t$  and receive feedback  $[y_t = \tilde{y}_t]$ 
9:   Compute  $\delta_t = \mathbf{1}_{\hat{y}_t} - \mathbf{1}_{\tilde{y}_t} \frac{[y_t = \tilde{y}_t]}{p_{\tilde{y}_t}}$  where  $\mathbf{1}_k$  stands for the vector with all its elements
      being zero except its  $k$ th element is 1.
10:  Compute  $W^t = W^{t-1} - \mathbf{x}_t \delta_t^\top$ 
11: end for
```

introduced here for the problem of online classification with bandit setting.

6.3 Balancing between Exploration and Exploitation

6.3.1 Preliminary

Algorithm 10 shows the Banditron algorithm [5], which is exactly the same algorithm as given in Algorithm 7 however it uses an adaptive γ_t to emphasize that the exploration vs. exploitation tradeoff parameter changes over time. Theorem 14 provides a new form for the mistake bound of Banditron. The proof of Theorem 14 is provided in Appendix A.16.

Theorem 14. *Let K be the number of classes. After running over a sequence of examples $\mathbf{x}_1, \dots, \mathbf{x}_T$, with $\|\mathbf{x}_t\|_2 \leq 1$ for all t , the expected number of mistakes made by Banditron, denoted by $\mathbb{E}[M]$, is bounded as follows*

$$\mathbb{E}[M] \leq |U|_F^2 + \sum_{t=1}^T \ell_t(U) + \mathbb{E} \left[\sum_{t=1}^T \gamma_t \right] + \mathbb{E} \left\{ \sum_{t=1}^T \gamma_t [\hat{y}_t = y_t] + \sum_{t=1}^T \frac{K}{\gamma_t} [\hat{y}_t \neq y_t] \right\} \quad (6.1)$$

where U is any arbitrary weight matrix (classifier) and $\{\gamma_t\}_{t=1}^T$ are exploration vs. exploitation tradeoff parameters of trials.

Remark: It is important to realize that by a proper re-scaling of the complexity of U and margin as suggested in [97], the bound provided in Theorem 14 can be rewritten as:

$$\mathbb{E}[M] \leq \sum_{t=1}^T \ell_t(U) + \mathbb{E} \left[\sum_{t=1}^T \gamma_t \right] + |U|_F \sqrt{2\mathbb{E} \left\{ \sum_{t=1}^T \gamma_t [\hat{y}_t = y_t] + \sum_{t=1}^T \frac{K}{\gamma_t} [\hat{y}_t \neq y_t] \right\}} \quad (6.2)$$

which is the inequality used in [5] to obtain the bound of Bandtiron given in Equation 5.2. More specifically, the bound in Equation 5.2 is obtained by two relaxations in Inequality 6.2: $[\hat{y}_t = y_t] \leq 1$ and $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. Moreover, notice that $\ell_t(U)$ is the hinge loss with margin equal to 1 in both inequalities 6.1 and 6.2.

Given the bound provided in Theorem 14, the optimal set of sampling probabilities $\{\gamma_t\}_{t=1}^T$ will be evidently obtained by minimizing the mistake bound stated in Theorem 14, i.e.

$$\mathcal{L} = \sum_{t=1}^T \frac{K}{\gamma_t} [\hat{y}_t \neq y_t] + \sum_{t=1}^T \gamma_t (1 + [\hat{y}_t = y_t])$$

However $[\hat{y}_t \neq y_t]$ and $[\hat{y}_t = y_t]$ are not provided as the feedback in the bandit setting and we need to approximate them with an expectation in terms of \tilde{y} . We consider the following approximations in Section 6.3.2 :

- $[\hat{y}_t = y_t] = \mathbb{E}_t \left[\frac{[\hat{y}_t = \tilde{y}_t][\tilde{y}_t = y_t]}{p_{\tilde{y}_t}} \right] \leq \mathbb{E}_t \left[\frac{[\hat{y}_t = \tilde{y}_t][\tilde{y}_t = y_t]}{1 - \gamma_t} \right] \leq 2\mathbb{E}_t [[\hat{y}_t = \tilde{y}_t][\tilde{y}_t = y_t]] = \mu_t$
- $[\hat{y}_t \neq y_t] = 1 - [\hat{y}_t = y_t] \leq 1 - \mathbb{E}_t [[\hat{y}_t = \tilde{y}_t][\tilde{y}_t = y_t]] = \tau_t$

To understand the merit of the above approximations, we analyze the following two approximations in Sections 6.3.3 and 6.3.4 respectively and use them as the competitors in the experiments.

- $[\hat{y}_t \neq y_t] \leq 1$ and $[\hat{y}_t = y_t] \leq \mu_t$
- $[\hat{y}_t = y_t] \leq 1$ (which is the relaxation used in [5]) and $[\hat{y}_t \neq y_t] \leq \tau_t$

6.3.2 Finding Optimal γ using $[\hat{y}_t \neq y_t] \leq \tau_t$ and $[\hat{y}_t = y_t] \leq \mu_t$

In order to bound the quantity $\hat{L} = \sum_{t=1}^T \frac{K}{\gamma_t} \tau_t + \sum_{t=1}^T \gamma_t (\mu + 1)$, we consider a general family of γ that is defined based on a concave function. We introduce the concept of *good support function*.

Definition 15. A function $\omega(z)$ defined in the domain of $z \geq 0$ is called a *good support function* if it satisfies the following conditions: (a) $\omega(z)$ is concave for $z \geq 0$ and $\omega(0) \geq 0$, (b) $\omega(z)$ is monotonically increasing, i.e., $\omega'(z) > 0$, for $z \geq 0$, (c) $\omega(z)$ is Lipschitz continuous with Lipschitz constant L , i.e., $\omega'(z) \leq L$, for $z \geq 0$, and (d) there exists a constant $\rho \geq 1$ such that for any $t \geq 0$ and $z \geq 0$, we have $\omega'(z) \leq \rho^t \omega'(z + t)$.

Proposition 6. (1) $\omega(z) = (a + z)^\lambda$, with $\lambda \in (0, 1]$ and $a > 0$, is a good support function, with Lipschitz constant $L = \lambda a^{\lambda-1}$, and $\rho = e^{(1-\lambda)/a}$, and (2) $\omega(z) = \ln(a + z)$ with $a > 0$ is a good support function, with $L = 1/a$, and $\rho = e^{1/a}$.

The proof of this Proposition is provided in Appendix A.17.

In order to bound quantity \hat{L} , we introduce two good support functions $\omega_1(z)$ and $\omega_2(z)$, with $\omega'_1(z) \geq \omega'_2(z)$ for any $z \geq 0$. We define

$$\gamma_t = \frac{\omega'_2\left(\sum_{i=1}^{t-1} 1 + \mu_i\right)}{2\omega'_1\left(\sum_{i=1}^{t-1} \tau_i\right)} \quad (6.3)$$

It is straightforward to verify that $\gamma_t \in (0, 1/2]$. In addition, since $\omega_1(z)$ and $\omega_2(z)$ are two concave functions, $\omega'_1(z)$ and $\omega'_2(z)$ are non-increasing functions of z , leading to a decreasing function of μ_t and t and increasing function of τ_t . The following proposition shows a key property for the construction of γ_t in 6.3. The proof is provided in Appendix A.18.

Proposition 7. *Given the construction of γ_t in (6.3), we have the following inequalities:*

$$\begin{aligned} \sum_{t=1}^T \gamma_t &\leq \rho_2 \frac{\omega_2(T)}{2\omega'_1\left(\sum_{t=1}^T \tau_t\right)} \quad , \quad \sum_{t=1}^T \gamma_t \mu_t \leq \rho_2^2 \frac{\omega_2\left(\sum_{t=1}^T \mu_t\right)}{2\omega'_1\left(\sum_{t=1}^T \tau_t\right)} \\ \sum_{t=1}^T \frac{K}{\gamma_t} \mu_t &\leq 2\rho_1 K \frac{\omega_1\left(\sum_{t=1}^T \tau_t\right)}{\omega'_2\left(\sum_{t=1}^T 1 + \mu_t\right)} \end{aligned}$$

where ρ_1 and ρ_2 are the constants defined in Definition 15 respectively for ω_1 and ω_2 .

Theorem 16. *Let $\omega_1(z)$ and $\omega_2(z)$ be two good support functions with $\omega'_1(z) \geq \omega'_2(z)$ for any $z \geq 0$. By running Algorithm 10 with γ_t set as in Eq. (6.3), we have the following bound for the expected number of misclassified examples*

$$\mathbb{E}[M] \leq \sum_{t=1}^T \ell_t(U) + \frac{\rho_2 \omega_2(T)}{2\omega'_1(T)} + |U|_F \left(\sqrt{\frac{2\rho_1 K \omega_1(3T)}{\omega'_2(3T)}} + \rho_2 \sqrt{\frac{\omega_2(2T)}{2\omega'_1(2T)}} \right)$$

where ρ_1 and ρ_2 are the constants of two good support functions.

Proof. The proof is straightforward by using the result in Remark I, Proposition 7, inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, and considering the fact that for a good support function ω : $\omega\left(\sum_{t=1}^T \tau_t\right) \leq \omega(T) \leq \omega(2T)$ and $\omega'\left(\sum_{t=1}^T \mu_t\right) \geq \omega'(2T)$. \square

Now, using the above theorem and Proposition 6, we have:

Corollary 17. *Suppose γ_t is in Eq. (6.3) with $\omega_1(z) = (1+z)^{\lambda_1}$ and $\omega_2(z) = (1+z)^{\lambda_2}$, where $\lambda_1, \lambda_2 \in (0, 1]$ and $\lambda_1 = \lambda_2 + 1/3$. By running Algorithm 10, we have the following bound for the expected number of misclassified examples*

$$\mathbb{E}[M] \leq \sum_{t=1}^T \ell_t(U) + \frac{\rho_2}{\lambda_1} (1+T)^{\frac{2}{3}} + |U|_F \left(\sqrt{\frac{\rho_2 e^{\frac{1}{3}} K}{\lambda_2} (1+3T)^{\frac{2}{3}}} + \frac{\rho_2}{\sqrt{\lambda_1}} (1+2T)^{\frac{1}{3}} \right)$$

where $\rho_2 = e^{1-\lambda_2}$. This bound is of $O(T^{2/3})$ and similar to the bound of the original Banditron.

Proof. It is a simple plug-in of the two support functions in Theorem 16. \square

6.3.3 Finding Optimal γ using $[\hat{y}_t \neq y_t] \leq 1$ and $[\hat{y}_t = y_t] \leq \mu_t$

In this section, we use the upper bound approximation $\hat{L} = \sum_{t=1}^T \frac{K}{\gamma_t} + \sum_{t=1}^T \gamma_t(1 + \mu_t)$.

Given a good support function $\omega(z)$, we define γ_t as

$$\gamma_t = \frac{1}{2L} \omega' \left(\sum_{i=1}^{t-1} 1 + \mu_i \right) \quad (6.4)$$

It is straightforward to see that γ_t is valid since $\gamma_t \in [0, 1/2]$. In addition, since $\omega(z)$ is a concave function, $\omega'(z)$ is a non-increasing function of z , leading to a decreasing value for γ_t as more and more training examples have been classified correctly. The following proposition shows a key property for the construction of γ_t in (6.4), with the proof provided in Appendix A.19.

Proposition 8. *Given the construction of γ_t in (6.4), we have the following inequalities*

$$\sum_{t=1}^T \gamma_t \mu_t \leq \frac{\rho^2}{2L} \omega \left(\sum_{t=1}^T \mu_t \right), \quad \sum_{t=1}^T \gamma_t \leq \frac{\rho}{2L} \omega(T), \quad \sum_{t=1}^T \frac{K}{\gamma_t} \leq \frac{2KLT}{\omega' \left(\sum_{t=1}^T 1 + \mu_t \right)}$$

Using the above proposition, we have the following theorem for the mistake bound of dynamic γ introduced in 6.4.

Theorem 18. *Let $\omega(z)$ be a good support function. By running Algorithm 10 with γ_t set as in Eq. (6.4), we have the following bound for the expected number of mistakes made by the algorithm*

$$\mathbb{E}[M] \leq \sum_{t=1}^T \frac{\rho \omega(T)}{2L} + |U|_F \left(\rho \sqrt{\frac{\omega(2T)}{2L}} + \sqrt{\frac{2KLT}{\omega'(3T)}} \right)$$

Proof. Similar to Theorem 16. \square

The following corollary directly follows from the result of Proposition 6 and Theorem 18.

Corollary 19. *By running Algorithm 10 with γ_t as in Eq. (6.4) and $\omega(z) = (1+z)^\lambda$, where $\lambda \in (0, 1]$, we have the following bound for the expected number of classification mistakes*

$$\mathbb{E}[M] \leq \sum_{t=1}^T \ell_t(U) + \frac{e^{1-\lambda}}{2\lambda} (1+T)^\lambda + |U|_F \left(\frac{e^{1-\lambda}}{\sqrt{2\lambda}} (1+2T)^{\frac{\lambda}{2}} + \sqrt{2K} (1+T)^{\frac{1-\lambda}{2}} T^{\frac{1}{2}} \right)$$

When $\lambda = 2/3$, we have $\mathbb{E}[M] = O(T^{2/3})$ which is the same convergence rate as Banditron.

6.3.4 Finding Optimal γ using $[\hat{y}_t \neq y_t] \leq \tau_t$ and $[\hat{y}_t = y_t] \leq 1$

Similar to the approach presented in the previous sections, we set γ_t as

$$\gamma_t = \frac{\omega'_2(t)}{2\omega'_1\left(\sum_{i=1}^{t-1} \tau_i\right)} \quad (6.5)$$

where $\omega_1(z)$ and $\omega_2(z)$ are two good support functions and $\omega'_1(z) \geq \omega'_2(z)$. It is easy to verify that $\gamma_t \in (0, 1/2]$ due to the properties of a good support function. The proposition below allows us to bound $\sum_{t=1}^T \gamma_t$ and $\sum_{t=1}^T K/\gamma_t$.

Proposition 9. *Given the construction of γ_t in (6.5), we have the following inequalities*

$$\sum_{t=1}^T \frac{K}{\gamma_t} \tau_t \leq 2K\rho \frac{\omega_1\left(\sum_{t=1}^T \tau_t\right)}{\omega'_2(T)} \quad \& \quad \sum_{t=1}^T \gamma_t \leq \frac{\omega_2(T)}{2\omega'_1\left(\sum_{t=1}^T \tau_t\right)}$$

Proof. Similar to Proposition 7. □

Theorem 20. *Let $\omega_1(z)$ and $\omega_2(z)$ be two good support functions. By running Algorithm 10 with γ_t set as in Eq. (6.5), we have the following bound for the number of misclassified*

examples $M = \sum_{t=1}^T [\hat{y}_t \neq y_t]$

$$\mathbb{E}[M] \leq \sum_{t=1}^T \ell_t(U) + \frac{\omega_2(T)}{2\omega'_1\left(\sum_{t=1}^T \tau_t\right)} + |U|_F \left(\sqrt{\frac{\omega_2(T)}{2\omega'_1(T)}} + \sqrt{2K\rho_1 \frac{\omega_1(T)}{\omega'_2(T)}} \right)$$

Proof. The proof directly follows Theorem 14 and Proposition 9. \square

The following corollary directly follows from the result of Proposition 6 and Theorem 20.

Corollary 21. *By running Algorithm 10 with γ_t in Eq. (6.5) and $\omega_1 = (1+z)^{\lambda_1}$ and $\omega_2 = (1+z)^{\lambda_2}$ with $\lambda_1, \lambda_2 \in (0, 1]$, we have the following bound for the expected number of misclassified examples*

$$\begin{aligned} \mathbb{E}[M] \leq \sum_{t=1}^T \ell_t(U) &+ \frac{1}{2\lambda_1}(1+T)^{\lambda_2+1-\lambda_1} \\ &+ |U|_F \left(\sqrt{\frac{1}{2\lambda_1}}(1+T)^{\frac{\lambda_2+1-\lambda_1}{2}} + \sqrt{\frac{2ke^{1-\lambda_1}}{\lambda_2}}(1+T)^{\frac{\lambda_1+1-\lambda_2}{2}} \right) \end{aligned}$$

with $\lambda_1 = \lambda_2 + \frac{1}{3}$ we have $\mathbb{E}[M] = O(T^{2/3})$ which is of the same rate as Banditron.

6.4 Experiments

In this section, we conduct experiments on the classification data sets, introduced in 1.6.1, to validate the proposed strategies for balancing the tradeoff between exploration and exploitation.

6.4.1 Experimental Settings

We refer to the algorithms developed in Sections 6.3.2, 6.3.3, and 6.3.4 as **banditron_ag3**, **banditron_ag1** and **banditron_ag2**. To evaluate the classification performance of the

three proposed learning strategies for exploitation vs. exploration tradeoff parameter γ , we compare them with three different version of Banditron, namely, **Banditron_worst**, **Banditron_Best**, and **Banditron_ag0**. **Banditron_worst** and **Banditron_Best** are Banditron algorithm when γ is set to the worst and best value for a given data. **Banditron_ag0** is the Banditron with the adaptive $\gamma_t = \frac{1}{2}t^{-1/3}$ as suggested in [5] for the general agnostic case. We repeat each experiment 50 times by generating random sequences of instances and report the average accumulate error rates, which are computed as the ratio of the number of misclassified samples to the number of samples received so far. For all three proposed methods in all the experiments, we use similar good support functions $\omega(z) = \omega_1(z) = \omega_2(z) = (1 + z)^\lambda$ with $\lambda = 0.1$ for a fair comparison. Also notice that the result is pretty stable for most of these data sets with different values of λ .

6.4.2 Experimental results

To study the behavior of different learning algorithms over trials, we show the average error rates of all the methods over the entire online process in Figure 6.2. First notice that there is big gap between **Banditron_worst** and **Banditron_Best** in all data sets that emphasizes that the Banditron algorithm can perform very poorly if γ is not set appropriately. We observe that overall the proposed algorithms exhibit similar or better learning rates as the Banditron algorithm with the optimal γ . In particular, **banditron_ag2** and **banditron_ag3** yields the best performance among the algorithms in comparison. In almost all the data sets, **banditron_ag2** and **banditron_ag3** perform significantly better than **banditron_ag0** which suggests that $\gamma_t = \frac{1}{2}t^{-1/3}$ is not a good adaptive choice. As a few examples, notice that the final error rate of **banditron_ag0** is 45% versus 38% error rate of **banditron_ag2**, **banditron_ag3** and **banditron_best** for MNIST data set. For Pendigits data set, the final error rate of **banditron_ag2** and **banditron_ag3** is 56% which is significantly low compared to 60% error rate of **banditron_best** and 62% error rate of **banditron_ag0**. The latter example also suggests that our adaptive strategy is better than the Banditron with a single best γ .

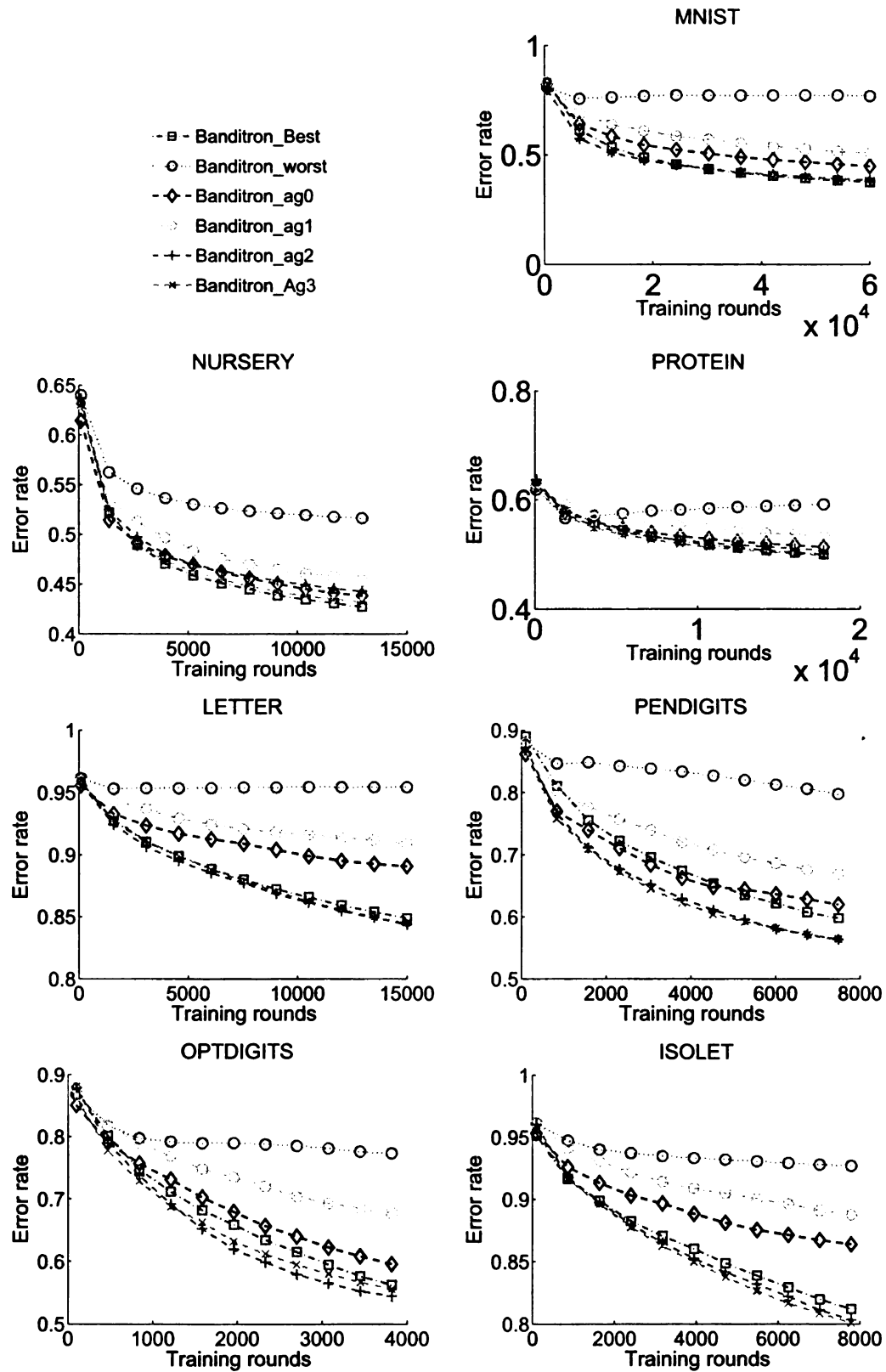


Figure 6.2: The error rates of different methods over trials. Each point on a curve is the average results of 50 randomly generated sequences of data.

Although better than `banditron_worst`, the performance of `banditron_ag1` is not comparable to that of the other methods. This can be explained by the inherited difference between `banditron_ag1` and the other two proposed approaches. Unlike `banditron_ag2` and `banditron_ag3` where two good support functions are introduced to determine γ_t , the γ_t defined in `banditron_ag1` is determined by a single good support function. As a result, we have a better control of the value for γ over time in `banditron_ag2` and `banditron_ag3` by a tradeoff between two functions: one which is the decreasing function of time and the other which is the increasing function of the number of misclassified examples.

Chapter 7

Conclusion and Future Work

In this chapter, we summarized the main contributions of this thesis and draw some directions for future work.

7.1 Summary and Conclusions

We developed several online and batch learning algorithms in this thesis. The batch learning algorithms that we covered have the common property that they all utilize boosting for optimizing an objective function in a function space. Utilizing boosting is particularly beneficial because it allows any existing supervised learning algorithms be applied for a new learning task. For the online learning, our focus has been on the classification with bandit feedback. In the following subsections, we briefly review our main contributions in two separate sections, one for boosting and one for online learning with bandit feedback.

7.1.1 Boosting

We developed boosting algorithms for several classification and ranking problems, as summarized below.

- **Semi-supervised classification:** Unlike existing semi-supervised learning algo-

rithms that focus on binary classification problems, we addressed the problem of multi-class semi-supervised learning directly. We proposed a new framework, termed multi-class semi-supervised boosting (MCSSB), that is able to improve the classification accuracy of any given base multi-class classifier. MCSSB utilizes both the cluster and manifold assumptions in the design of objective function and exploits boosting techniques to optimize the objective function. We showed that our proposed framework is able to improve the performance of a given classifier much better than Assemble, a well-known semi-supervised boosting algorithm, on several real world data sets. We also showed that MCSSB is very robust to the choice of base classifiers, the number of labeled examples, and the value of parameter C .

- **Learning to rank by maximizing NDCG:** Listwise approach is a relatively new approach to learning to rank that aims to optimize listwise loss functions; i.e. loss functions that measure the performance of a ranking model in the query-level. The difficulty in optimizing such losses lies in the inherited sort function used for computing them. We address this challenge by a probabilistic framework for the problem of maximizing NDCG that optimizes the expectation of NDCG over all the possible permutations of documents. We present a relaxation strategy to effectively approximate the expectation of NDCG, and a bound optimization strategy for efficient optimization. Our experiments on benchmark data sets shows that our method is superior to the state-of-the-art learning to rank algorithms in terms of performance and stability.
- **Ranking Refinement:** We considered the problem of ranking refinement whose goal is to improve a given ranking function by a small number of labeled instances. The key challenge in combining the ranking information from the base ranker and the labeled instances arises from the fact that the information in the base ranker tends to be inaccurate and the information from the training data tends to be noisy. We presented a multiplicative objective function to combine these sources of information

and proposed a boosting algorithm for learning. Empirical studies with relevance feedback and recommender system show promising performance of the proposed algorithm.

7.1.2 Online Learning

- **General framework:** We presented a general framework for online multi-class learning with partial feedback using the potential-based gradient descent approach of which Banditron is a special case. In addition, we proposed an exponential gradient algorithm for online multi-class learning with partial feedback. Compared to the Banditron algorithm, the exponential gradient algorithm is advantageous in that its mistake bound is independent from the dimension of data, making it suitable for classifying high dimensional data. We verified the efficacy of the proposed algorithm by empirical studies with several real-world data sets. Our experiments show the exponential gradient approach for online learning with partial feedback is more effective than Banditron in terms of the learning rate, which makes it more suitable for the scenario when the number of training examples is relatively small.
- **Automatic tuning of trade-off parameter :** We studied the problem of optimizing the exploration-exploitation tradeoff in the context of online classification with bandit feedback. We proposed three different strategies to automatically tune the tradeoff parameter used by the Banditron algorithm. We showed through extensive experimental study that the proposed approaches are effective in adjusting the exploration-exploitation tradeoff. In particular, we found that two of the proposed algorithms achieve similar or better performance compared to Banditron with the best value for γ .

7.2 Future Work

In this section, we summarize future research directions that are directly related to the theme of this thesis, in two separate subsections, one for boosting and one for online learning.

7.2.1 Boosting

There has recently been increasing interests in understanding the relation between game theory and machine learning and furthermore examining how each field contributes to the other [98, 99]. Particularly, boosting can be considered a fictitious zero-sum game [39] between two agents: a data generator as a row player that chooses a mixed strategy over the space of training examples and a learner as a column player that chooses strategies over the hypothesis space. The followings are some interesting game theory questions for boosting:

- **Representability of a given hypothesis for an specific task:** Using Minimax theorem, Freund et al. [39] showed that there is a mixed strategy over the space of hypotheses H that produces zero classification error over the training set if for any mixed strategy over the training examples, there is one hypothesis in H able to perform better than random guessing. Similar results may be extended to other tasks that also utilize boosting. For example, we utilized the space of binary classifiers to learn a ranking algorithm that maximizes NDCG in Chapter 3. It is interesting to study the ability and limitation of binary hypotheses in maximizing NDCG; i.e. to analyze the maximum value of NDCG obtained by a mixed strategy over the binary hypotheses given.
- **New methods to find mixed strategies:** Boosting (and other ensemble methods) can be considered methods to find the mixed strategy over the hypothesis space. However, the designer of these methods did not have the notion of equilibrium in

mind while developing them. Designing new algorithms that directly consider the data generator as the row player and the learner and the column player and aim to find an equilibrium solution is potentially advantageous and interesting. One possible option is to learn a finite set of weak models sequentially (similar to boosting) and then playing a game to find the best weighted majority votes (mixed strategy).

- **Batch learning with partial feedback:** In this problem, the feedback (i.e. labeling) is similar to online learning with partial feedback except that training instances are provided in batch mode. For instance, consider the multi-class learning problem where each instance is given a class label and a flag that indicates whether or not the given class label is correct. Similar to online learning with partial feedback, contextual advertisement and recommender systems are some example applications of this problem. For these problems, training examples can be collected and utilized for learning in batch mode similar to the click-through ranking feedback that is being used in learning to rank. Designing a boosting algorithm that utilizes a supervised classifier for this problem is one direction of research work. From the game theory point of view, this problem can be considered a game between two players with partially known payoff matrix.

7.2.2 Online learning

Online learning with bandit feedback is a new research area for which there are several open research questions, as summarized below:

- **Tighter bounds:** Kakade et. al [5] proved that there exists algorithms for online classification with bandit feedback with bounds of order $O(T^{1/2})$, however the algorithms that are introduced so far are of order $O(T^{2/3})$. Developing algorithms that have better regret bounds than existing ones is one of the future research directions.
- **Online learning to rank and multi-label classification with partial feedback:** Contextual advertising and recommender systems are originally ranking problems

that were simplified as multi-class problems when dealing with online partial feedback. An intermediate setting between online ranking and online classification with bandit setting is online multi-label classification in which more than one class (advertisement) are relevant. Developing algorithms for online learning to rank and online multi-label classification with partial feedback is another research direction that will be explored in the future.

APPENDICES

Appendix A

APPENDIX

A.1 Proof of Lemma 1, Chapter 2

Proof. Bound in Equation (2.8) can be derived as follows:

$$\begin{aligned}
 \frac{1}{\tilde{Z}_{i,j}^u} &= \frac{1}{\sum_{k'=1}^m \tilde{b}_i^k \tilde{b}_j^k} = \frac{(\sum_{k'=1}^m b_i^{k'} \exp(\alpha h_i^{k'})) (\sum_{k'=1}^m b_j^{k'} \exp(\alpha h_j^{k'}))}{\sum_{k=1}^m b_i^k b_j^k \exp(\alpha(h_i^k + h_j^k))} \\
 &\leq \left(\sum_{k'=1}^m b_i^{k'} \exp(\alpha h_i^{k'}) \right) \left(\sum_{k'=1}^m b_j^{k'} \exp(\alpha h_j^{k'}) \right) \times \frac{\left(\sum_{k=1}^m \tau_{i,j}^k \exp(-\alpha(h_i^k + h_j^k)) \right)}{\sum_{k=1}^m b_i^k b_j^k} \\
 &= \sum_{k_1, k_2, k_3=1}^m \frac{b_i^{k_1} b_j^{k_2} \tau_{i,j}^{k_3}}{Z_{i,j}^u} \exp \left(\alpha(h_i^{k_1} + h_j^{k_2} - h_i^{k_3} - h_j^{k_3}) \right) \\
 &\leq \frac{1 + \exp(6\alpha) + \exp(-6\alpha)}{3Z_{i,j}^u} - \frac{\exp(6\alpha) - 1}{3Z_{i,j}^u} \left(\sum_{k=1}^m (\tau_{i,j}^k - b_i^k) h_i^k \right) \tag{A.1}
 \end{aligned}$$

The inequality in (A.1) follows the convexity of reciprocal function, i.e.,

$$\begin{aligned}
 \frac{1}{\sum_{k=1}^m b_i^k b_j^k \exp(\alpha(h_i^k + h_j^k))} &= \frac{1}{\sum_{k=1}^m b_i^k b_j^k} \frac{1}{\sum_{k=1}^m \tau_{i,j}^k \exp(\alpha(h_i^k + h_j^k))} \\
 &\leq \frac{1}{\sum_{k=1}^m b_i^k b_j^k} \sum_{k=1}^m \tau_{i,j}^k \exp(-\alpha(h_i^k + h_j^k))
 \end{aligned}$$

The inequality in (A.1) follows the convexity of exponential function, i.e.,

$$\begin{aligned} \exp(\alpha(h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3})) &= \exp\left(6\alpha \frac{h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3} + 2}{6} + 0 \times \frac{-h_i^{k1} - h_j^{k2} + h_i^{k3} + h_j^{k3} + 2}{6} + 6\alpha \frac{1}{3}\right) \\ &\leq \frac{h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3} + 2}{6} \exp(6\alpha) + \frac{1}{3} \exp(6\alpha) + \frac{-h_i^{k1} - h_j^{k2} + h_i^{k3} + h_j^{k3} + 2}{6} \end{aligned}$$

Bound in Equation 2.9 can be derived as follows

$$\begin{aligned} \frac{1}{\bar{Z}_{i,j}^l} &= \sum_{k'=1}^m y_i^k \exp(H_j^{k'} - H_j^k + \alpha(h_j^{k'} - h_j^k)) \\ &\leq \frac{1 + \exp(6\alpha) + \exp(-6\alpha)}{3Z_{i,j}^l} + \frac{\exp(6\alpha) - 1}{6} \sum_{k=1}^m h_i^k \left(\sum_{k'=1}^m y_j^{k'} \frac{b_i^k}{b_i^{k'}} - \frac{y_i^k}{b_i^k} \right) \end{aligned}$$

The inequality used by the above derivation follows the convexity of exponential function, i.e.,

$$\begin{aligned} \exp(\alpha(h_i^{k'} - h_j^k)) &\leq \exp\left(6\alpha \frac{h_i^{k'} - h_j^k + 2}{6} + 0 \times \frac{-h_i^{k'} + h_j^k + 2}{6} + 6\alpha \frac{1}{3}\right) \\ &\leq \frac{h_i^{k'} - h_j^k + 2}{6} \exp(6\alpha) + \frac{1}{3} \exp(6\alpha) + \frac{-h_i^{k'} + h_j^k + 2}{6} \end{aligned}$$

Using the definition of $\phi_{i,j}^k$, we have the result in Equation 2.9. □

A.2 Proof of Lemma 2, Chapter 2

Proof. Following the result in (A.1), we have

$$\begin{aligned} \frac{1}{\bar{Z}_{i,j}^u} \sum_{k_1, k_2, k_3=1}^m \frac{b_i^{k1} b_j^{k2} \tau_{i,j}^{k3}}{Z_{i,j}^u} \exp(\alpha(h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3})) \\ \leq \frac{1}{Z_{i,j}^u} + \frac{\exp(2\alpha) - 1}{2Z_{i,j}^u} \left(\sum_{k=1}^m h_i^k b_i^k + h_j^k b_j^k \right) + \frac{\exp(-2\alpha) - 1}{2Z_{i,j}^u} \left(\sum_{k=1}^m a_{i,j}^k (h_i^k + h_j^k) \right) \end{aligned}$$

The inequality in (A.1) follows the convexity of exponential function, i.e.,

$$\begin{aligned} \exp(\alpha(h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3})) &= \exp \left(6\alpha \frac{h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3} + 2}{6} + 0 \times \frac{-h_i^{k1} - h_j^{k2} + h_i^{k3} + h_j^{k3} + 2}{6} + 6\alpha \frac{1}{3} \right) \\ &\leq \frac{h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3} + 2}{6} \exp(6\alpha) + \frac{1}{3} \exp(6\alpha) + \frac{-h_i^{k1} - h_j^{k2} + h_i^{k3} + h_j^{k3} + 2}{6} \end{aligned}$$

Bound in Equation 2.9 can be derived as follows

$$\begin{aligned} \frac{1}{\tilde{Z}_{i,j}^l} &= \sum_{k',k=1}^m y_i^k \exp(H_j^{k'} - H_j^k + \alpha(h_j^{k'} - h_j^k)) \\ &\leq \frac{1 + \exp(6\alpha) + \exp(-6\alpha)}{3Z_{i,j}^l} + \frac{\exp(6\alpha) - 1}{6} \sum_{k=1}^m h_i^k \left(\sum_{k'=1}^m y_j^{k'} \frac{b_i^k}{b_i^{k'}} - \frac{y_i^k}{b_i^k} \right) \end{aligned}$$

The inequality used by the above derivation follows the convexity of exponential function, i.e.,

$$\begin{aligned} \exp(\alpha(h_i^{k'} - h_j^k)) &\leq \exp \left(6\alpha \frac{h_i^{k'} - h_j^k + 2}{6} + 0 \times \frac{-h_i^{k'} + h_j^k + 2}{6} + 6\alpha \frac{1}{3} \right) \\ &\leq \frac{h_i^{k'} - h_j^k + 2}{6} \exp(6\alpha) + \frac{1}{3} \exp(6\alpha) + \frac{-h_i^{k'} + h_j^k + 2}{6} \end{aligned}$$

Using the definition of $\phi_{i,j}^k$, we have the result in Equation 2.9. □

A.2 Proof of Lemma 2, Chapter 2

Proof. Following the result in (A.1), we have

$$\begin{aligned} \frac{1}{\tilde{Z}_{i,j}^u} \sum_{k_1, k_2, k_3=1}^m \frac{b_i^{k1} b_j^{k2} \tau_{i,j}^{k3}}{Z_{i,j}^u} \exp(\alpha(h_i^{k1} + h_j^{k2} - h_i^{k3} - h_j^{k3})) \\ \leq \frac{1}{Z_{i,j}^u} + \frac{\exp(2\alpha) - 1}{2Z_{i,j}^u} \left(\sum_{k=1}^m h_i^k b_i^k + h_j^k b_j^k \right) + \frac{\exp(-2\alpha) - 1}{2Z_{i,j}^u} \left(\sum_{k=1}^m a_{i,j}^k (h_i^k + h_j^k) \right) \end{aligned}$$

The inequality in (A.1) follows the convexity of exponential function, i.e.,

$$\begin{aligned} \exp(\alpha(h_i^{k_1} + h_j^{k_2} - h_i^{k_3} - h_j^{k_3})) &= \exp \left(6\alpha \frac{h_i^{k_1} + h_j^{k_2} - h_i^{k_3} - h_j^{k_3} + 2}{6} + 0 \times \frac{-h_i^{k_1} - h_j^{k_2} + h_i^{k_3} + h_j^{k_3} + 2}{6} + 6\alpha \frac{1}{3} \right) \\ &\leq \frac{h_i^{k_1} + h_j^{k_2} - h_i^{k_3} - h_j^{k_3} + 2}{6} \exp(6\alpha) + \frac{1}{3} \exp(6\alpha) + \frac{-h_i^{k_1} - h_j^{k_2} + h_i^{k_3} + h_j^{k_3} + 2}{6} \end{aligned}$$

Bound in Equation 2.9 can be derived as follows

$$\begin{aligned} \frac{1}{\tilde{Z}_{i,j}^l} &= \sum_{k',k=1}^m y_i^k \exp(H_j^{k'} - H_j^k + \alpha(h_j^{k'} - h_j^k)) \\ &\leq \frac{1 + \exp(6\alpha) + \exp(-6\alpha)}{3Z_{i,j}^l} + \frac{\exp(6\alpha) - 1}{6} \sum_{k=1}^m h_i^k \left(\sum_{k'=1}^m y_j^{k'} \frac{b_i^k}{b_i^{k'}} - \frac{y_i^k}{b_i^k} \right) \end{aligned}$$

The inequality used by the above derivation follows the convexity of exponential function, i.e.,

$$\begin{aligned} \exp(\alpha(h_i^{k'} - h_j^k)) &\leq \exp \left(6\alpha \frac{h_i^{k'} - h_j^k + 2}{6} + 0 \times \frac{-h_i^{k'} + h_j^k + 2}{6} + 6\alpha \frac{1}{3} \right) \\ &\leq \frac{h_i^{k'} - h_j^k + 2}{6} \exp(6\alpha) + \frac{1}{3} \exp(6\alpha) + \frac{-h_i^{k'} + h_j^k + 2}{6} \end{aligned}$$

Using the definition of $\phi_{i,j}^k$, we have the result in Equation 2.9. □

A.2 Proof of Lemma 2, Chapter 2

Proof. Following the result in (A.1), we have

$$\begin{aligned} \frac{1}{\tilde{Z}_{i,j}^u} \sum_{k_1, k_2, k_3=1}^m \frac{b_i^{k_1} b_j^{k_2} \tau_{i,j}^{k_3}}{Z_{i,j}^u} \exp(\alpha(h_i^{k_1} + h_j^{k_2} - h_i^{k_3} - h_j^{k_3})) \\ \leq \frac{1}{Z_{i,j}^u} + \frac{\exp(2\alpha) - 1}{2Z_{i,j}^u} \left(\sum_{k=1}^m h_i^k b_i^k + h_j^k b_j^k \right) + \frac{\exp(-2\alpha) - 1}{2Z_{i,j}^u} \left(\sum_{k=1}^m a_{i,j}^k (h_i^k + h_j^k) \right) \end{aligned}$$

The inequality in the above derivation follows the convexity of exponential function (similar to the proof of Lemma 1). For $Z_{i,j}^l$, we have

$$\begin{aligned} \frac{1}{\tilde{Z}_{i,j}^l} &= \sum_{k,k'=1}^m y_i^k \frac{b_j^{k'}}{b_j^k} \exp \left(2\alpha \frac{h_j^{k'}}{2} - 2\alpha \frac{h_j^k}{2} + 0 \frac{2 - h_j^{k'} - h_j^k}{2} \right) \\ &\leq \sum_{k,k'=1}^m y_i^k \frac{b_j^{k'}}{b_j^k} \left(\frac{h_j^{k'}}{2} \exp(2\alpha) + \frac{h_j^k}{2} \exp(-2\alpha) \right) + \sum_{k,k'=1}^m y_i^k \frac{b_j^{k'}}{b_j^k} \frac{2 - h_j^{k'} - h_j^k}{2} \end{aligned}$$

Replacing $1/\tilde{Z}_{i,j}^u$ and $1/\tilde{Z}_{i,j}^l$ in (2.8) and (2.9) with the above bounds, we have the result in Lemma 2. \square

A.3 Proof of Theorem 4, Chapter 2

Using Lemma 2 and Theorem 3, we have

$$\begin{aligned} \tilde{F} - F &\leq \sqrt{\frac{B_u + CB_l}{A_u + CA_l}} (A_u + CA_l) + \sqrt{\frac{A_u + CA_l}{B_u + CB_l}} (B_u + CB_l) - (A_u + CA_l + B_u + CB_l) \\ &= - \left(\sqrt{A_u + CA_l} - \sqrt{B_u + CB_l} \right)^2, \end{aligned}$$

which is equivalent to

$$\begin{aligned} \frac{\tilde{F}}{F} &\leq 1 - \frac{(\sqrt{A_u + CA_l} - \sqrt{B_u + CB_l})^2}{F} \\ &\leq \exp \left(- \frac{(\sqrt{A_u + CA_l} - \sqrt{B_u + CB_l})^2}{F} \right) \end{aligned} \tag{A.2}$$

The above inequality follows from $\exp(x) \geq 1 + x$. We rewrite F^T as

$$F^T = F^0 \prod_{t=1}^T (F^t / F^{t-1})$$

By substituting F^t / F^{t-1} with the bound in Equation A.6, we have the result in the theorem.

A.4 Proof of Proposition 2, Chapter 3

$$\begin{aligned}
& \frac{1}{1 + \exp(\tilde{F}_i^k - \tilde{F}_j^k)} = \frac{1}{1 + \exp(F_i^k - F_j^k + \alpha(f_i^k - f_j^k))} \\
&= \frac{1}{1 + \exp(F_i^k - F_j^k)} \left(\frac{1}{1 + \exp(F_i^k - F_j^k)} + \frac{\exp(F_i^k - F_j^k)}{1 + \exp(F_i^k - F_j^k)} \exp(\alpha(f_i^k - f_j^k))^{-1} \right) \\
&\leq \frac{1}{1 + \exp(F_i^k - F_j^k)} \left(1 - \frac{\exp(F_i^k - F_j^k)}{1 + \exp(F_i^k - F_j^k)} + \frac{\exp(F_i^k - F_j^k)}{1 + \exp(F_i^k - F_j^k)} \exp(\alpha(f_j^k - f_i^k)) \right) \\
&= \frac{1}{1 + \exp(F_i^k - F_j^k)} + \gamma_{i,j}^k [\exp(\alpha(f_j^k - f_i^k)) - 1]
\end{aligned}$$

The first step is a simple manipulations of the terms and the second step is due to the convexity of inverse function on R^+ .

A.5 Proof of Lemma 4, Chapter 3

$$\begin{aligned}
1 &= \sum_{\pi^k \in G_a^k(i,j)} \Pr(\pi^k | F, q^k) + \sum_{\pi^k \in G_b^k(i,j)} \Pr(\pi^k | F, q^k) \\
&= \sum_{\pi^k \in G_a^k(i,j)} \Pr(\pi^k | F, q^k) \left(1 + \exp \left[2(\pi^k(i) - \pi^k(j))(F(d_i^k, q^k) - F(d_j^k, q^k)) \right] \right) \\
&\geq \sum_{\pi^k \in G_a^k(i,j)} \left(\Pr(\pi^k | F, q^k) \left(1 + \exp \left[2(F(d_i^k, q^k) - F(d_j^k, q^k)) \right] \right) \right) \\
&= \left(1 + \exp \left[2(F(d_i^k, q^k) - F(d_j^k, q^k)) \right] \right) \Pr(\pi^k(i) > \pi^k(j))
\end{aligned}$$

We used the definition of $\Pr(\pi^k | F, q^k)$ in Equation (3.6) to find $G_b^k(i, j)$ as the dual of $G_a^k(i, j)$ in the first step of the proof. The inequality in the proof is because $\pi^k(i) - \pi^k(j) \geq 1$ and the last step is because $\Pr(\pi^k | F, q^k)$ is the only term dependent on π .

A.6 Proof of Theorem 5, Chapter 3

In order to obtain the result of the Theorem 5, we first plug Equation (3.13) in Equation (3.11). This leads to minimizing $\sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k [\exp(\alpha(f_j^k - f_i^k))]$, the term related to α . Since f_i^k takes binary values 0 and 1, we have the following:

$$\begin{aligned} & \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k \exp(\alpha(f_j^k - f_i^k)) \\ &= \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k \left(\exp(\alpha) I(f_j^k > f_i^k) + \exp(-\alpha) I(f_j^k < f_i^k) \right) \end{aligned}$$

Getting the partial derivative of this term respect to α and having it equal to zero results the theorem.

A.7 Proof of Theorem 6, Chapter 3

First, we provide the following proposition to handle $\exp(\alpha(f_j^k - f_i^k))$.

Proposition 10. *If $x, y \in [0, 1]$, we have*

$$\exp(\alpha(x - y)) \leq \frac{\exp(3\alpha) - 1}{3} (x - y) + \frac{\exp(3\alpha) + \exp(-3\alpha) + 1}{3} \quad (\text{A.3})$$

Proof. Due to the convexity of \exp function, we have:

$$\begin{aligned} \exp(\alpha(x - y)) &= \exp\left(3\alpha \frac{x - y + 1}{3} + 0 \times \frac{1 - x + y}{3} + \frac{1}{3} \times -3\alpha\right) \\ &\leq \frac{x - y + 1}{3} \exp(3\alpha) + \frac{1 - x + y}{3} + \frac{1}{3} \exp(-3\alpha) \end{aligned}$$

□

Using the result in the above proposition, we can bound the last term in Equation (3.13) as follows:

$$\theta_{i,j}^k [\exp(\alpha(f_j^k - f_i^k)) - 1] \leq \theta_{i,j}^k \left(\frac{\exp(3\alpha) - 1}{3} (f_j^k - f_i^k) + \frac{\exp(3\alpha) + \exp(-3\alpha) - 2}{3} \right) \quad (\text{A.4})$$

Using the result in Equation (A.4) and (3.13), we have $\bar{\mathcal{M}}(Q, \tilde{F})$ in Equation (3.11) bounded as

$$\begin{aligned}\bar{\mathcal{M}}(Q, \tilde{F}) &\leq \bar{\mathcal{M}}(Q, F) + \gamma(\alpha) + \frac{\exp(3\alpha) - 1}{3} \sum_{k=1}^n \sum_{i=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \sum_{j=1}^{m_k} \theta_{i,j}^k (f_i^k - f_j^k) \\ &= \bar{\mathcal{M}}(Q, F) + \gamma(\alpha) + \frac{\exp(3\alpha) - 1}{3} \sum_{k=1}^n \sum_{i=1}^{m_k} f_i^k \left(\sum_{j=1}^{m_k} \frac{2^{r_i^k} - 2^{r_j^k}}{Z_k} \theta_{i,j}^k \right)\end{aligned}$$

A.8 Proof of Theorem 7, Chapter 3

Proof. By plugging Equation 13 into Equation 11, we have

$$\bar{\mathcal{M}}(Q, \tilde{F}_i^k) - \bar{\mathcal{M}}(Q, F) \leq \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k \left[\exp(\alpha(f_j^k - f_i^k)) - 1 \right]$$

Since f_i^k takes binary values 0 and 1, we have the following

$$\begin{aligned}& \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \left[\theta_{i,j}^k \exp(\alpha(f_j^k - f_i^k)) - 1 \right] \\ &= \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k \left(\exp(\alpha) I(f_j^k > f_i^k) + \exp(-\alpha) I(f_j^k < f_i^k) \right) \\ &- \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k \left(I(f_j^k > f_i^k) + I(f_j^k < f_i^k) + I(f_j^k = f_i^k) \right)\end{aligned}$$

So,

$$\begin{aligned}
\bar{\mathcal{M}}(Q, \tilde{F}_i^k) - \bar{\mathcal{M}}(Q, F) &\leq \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k \left(\exp(\alpha) I(f_j^k > f_i^k) + \exp(-\alpha) I(f_j^k < f_i^k) \right) \\
&- \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k \left(I(f_j^k > f_i^k) + I(f_j^k < f_i^k) + I(f_j^k = f_i^k) \right) \\
&\leq \exp(\alpha) \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k I(f_j^k > f_i^k) \\
&+ \exp(-\alpha) \sum_{k=1}^n \sum_{i,j=1}^{m_k} \frac{2^{r_i^k} - 1}{Z_k} \theta_{i,j}^k I(f_j^k < f_i^k) - \alpha_1 - \alpha_2 \\
&= \exp(\alpha) \alpha_2 + \exp(-\alpha) \alpha_1 - \alpha_1 - \alpha_2 \\
&= \exp\left(\frac{1}{2} \log\left(\frac{\alpha_1}{\alpha_2}\right)\right) \alpha_2 + \exp\left(-\frac{1}{2} \log\left(\frac{\alpha_1}{\alpha_2}\right)\right) \alpha_1 - \alpha_1 - \alpha_2 \\
&= \alpha_2 \sqrt{\frac{\alpha_1}{\alpha_2}} + \alpha_1 \sqrt{\frac{\alpha_2}{\alpha_1}} - \alpha_1 - \alpha_2 = -(\sqrt{\alpha_1} - \sqrt{\alpha_2})^2
\end{aligned}$$

which is equivalent to

$$\frac{\bar{\mathcal{M}}(Q, \tilde{F}_i^k)}{\bar{\mathcal{M}}(Q, F)} \leq 1 - \frac{(\sqrt{\alpha_1} - \sqrt{\alpha_2})^2}{\bar{\mathcal{M}}(Q, F)} \quad (\text{A.5})$$

$$\leq \exp\left(-\frac{(\sqrt{\alpha_1} - \sqrt{\alpha_2})^2}{\bar{\mathcal{M}}(Q, F)}\right) \quad (\text{A.6})$$

The above inequality follows from $\exp(x) \geq 1 + x$. We rewrite $\bar{\mathcal{M}}^T$ as

$$\bar{\mathcal{M}}^T = \bar{\mathcal{M}}^0 \prod_{t=1}^T \frac{\bar{\mathcal{M}}^t}{\bar{\mathcal{M}}^{t-1}}$$

By substituting $\frac{\bar{\mathcal{M}}^t}{\bar{\mathcal{M}}^{t-1}}$ with the bound in Equation A.6, we have the result in the theorem. \square

A.9 Proof of Theorem 8, Chapter 4

Proof. First, note that the objective function L_p is convex in terms of \mathbf{F} . This is because L_p can be expanded as follows:

$$L_p = \sum_{i,j,k,l=1}^n T_{i,j} W_{i,j} \exp(F_j - F_i + F_k - F_l)$$

Since $\exp(F_j - F_i + F_k - F_l)$ is a convex function, L_p is convex. Since L_p is a convex function, the solution found by minimizing L_p will always be global optimal, instead of local optimal.

Second, to show that the optimal solution found by minimizing L_p is Pareto efficient, we prove by contradiction. Let \mathbf{F}^* denote the global minimizer of function L_p . By assuming that Theorem 8 is not correct, there will exist a solution $\mathbf{F} \neq \mathbf{F}^*$ that either (1) $\widehat{err}_w(\mathbf{F}) < \widehat{err}_w(\mathbf{F}^*)$ and $\widehat{err}_t(\mathbf{F}) \leq \widehat{err}_t(\mathbf{F}^*)$, or (2) $\widehat{err}_w(\mathbf{F}) \leq \widehat{err}_w(\mathbf{F}^*)$ and $\widehat{err}_t(\mathbf{F}) < \widehat{err}_t(\mathbf{F}^*)$. We can easily infer $L_p(\mathbf{F}) < L_p(\mathbf{F}^*)$ since (1) both \widehat{err}_w and \widehat{err}_t are non-negative for any solution \mathbf{F} , and (2) $L_p = \widehat{err}_w \times \widehat{err}_t$. Clearly, this conclusion contradicts the fact that \mathbf{F}^* is a global minimizer of L_p . \square

A.10 Proof of Lemma 5, Chapter 4

Proof. Since $\tilde{F}(\mathbf{x}) = F(\mathbf{x}) + \alpha f(\mathbf{x})$, we have

$$\begin{aligned} \frac{\tilde{L}_p}{L_p} &= \left(\sum_{i,j=1}^n W_{i,j} \exp(F_j - F_i + \alpha(f_j - f_i)) \right) \times \left(\sum_{i,j=1}^n T_{i,j} \exp(F_j - F_i + \alpha(f_j - f_i)) \right) \\ &= \left(\sum_{i,j=1}^n a_{i,j} \exp(\alpha(f_j - f_i)) \right) \left(\sum_{i,j=1}^n b_{i,j} \exp(\alpha(f_j - f_i)) \right) \end{aligned}$$

where $a_{i,j}$ and $b_{i,j}$ are defined in (4.11) and (4.12). Thus, we have an upper bound of the log ratio as follows

$$\begin{aligned} \log \frac{\tilde{L}_p}{L} &= \log \left(\sum_{i,j=1}^n a_{i,j} \exp(\alpha(f_j - f_i)) \right) + \log \left(\sum_{i,j=1}^n b_{i,j} \exp(\alpha(f_j - f_i)) \right) \\ &\leq -2 + \sum_{i,j=1}^n (a_{i,j} + b_{i,j}) \exp(\alpha(f_j - f_i)) \end{aligned}$$

The second inequality follows the concaveness of the logarithm function, i.e., $\log x \leq x - 1$ for any $x > 0$. □

A.11 Proof of Theorem 9, Chapter 4

Proof. Using the upper bound expressed in Lemma 5, we have

$$\begin{aligned} \log \frac{\tilde{L}_p}{L_p} + 2 &\leq \sum_{i,j=1}^n \gamma_{i,j} \exp(\alpha(f_j - f_i)) \\ &= \left(\sum_{i,j=1}^n \gamma_{i,j} \delta(f_j, 1) \delta(f_i, 0) \right) \exp(\alpha) + \left(\sum_{i,j=1}^n \gamma_{i,j} \delta(f_j, 0) \delta(f_i, 1) \right) \exp(-\alpha) \end{aligned}$$

Using the definition of α in (8), we have

$$\begin{aligned} \log \frac{\tilde{L}_p}{L_p} &\leq -2 + 2 \sqrt{\left(\sum_{i,j=1}^n \gamma_{i,j} \delta(f_j, 1) \delta(f_i, 0) \right) \left(\sum_{i,j=1}^n \gamma_{i,j} \delta(f_j, 0) \delta(f_i, 1) \right)} \\ &= -2 + 2\sqrt{\mu\nu} \end{aligned}$$

In the above, we use the definitions of μ and ν in Theorem 8 to simplify the expression. Since $2 = \sum_{i,j=1}^n \gamma_{i,j} \geq \mu + \nu$, we have

$$\log \frac{\tilde{L}_p}{L_p} \leq -2 + 2\sqrt{\mu\nu} \leq -\mu - \nu + 2\sqrt{\mu\nu} = -(\sqrt{\mu} - \sqrt{\nu})^2$$

We thus have

$$\log \frac{L_p^t}{L_p^{t-1}} \leq r_t = -(\sqrt{\mu_t} - \sqrt{\nu_t})^2$$

Substituting the above expression for r_t into (4.17), and further using the fact

$$L_0 = \sum_{i,j=1}^n T_{i,j} + W_{i,j},$$

we obtain the result in Theorem 9. □

A.12 Proof of Theorem 10, Chapter 4

Proof. We rewrite the quantity η as follows:

$$\eta = \sum_{i=1}^n f_i y_i |w_i| = \sum_{i=1}^n f_i \left(\sum_{j=1}^n \gamma_{i,j} - \gamma_{j,i} \right) = \sum_{i,j=1}^n \gamma_{i,j} (f_i - f_j) = \mu - \nu$$

Since

$$\mu - \nu = (\sqrt{\mu} - \sqrt{\nu}) (\sqrt{\mu} + \sqrt{\nu}) \geq (\sqrt{\mu} - \sqrt{\nu})^2,$$

we have $\eta \geq (\sqrt{\mu} - \sqrt{\nu})^2$. Substituting this result into the expression of Theorem 9, we have Theorem 10. □

A.13 Proof of Proposition 5, Chapter 5

Proof.

$$\begin{aligned}
\mathbb{E}_t[|\delta_t|_s^2] &= \mathbb{E}_t \left[\left| \mathbf{1}_{\hat{y}_t} - \mathbf{1}_{\tilde{y}_t} \frac{[y_t = \tilde{y}_t]}{p_{\tilde{y}_t}} \right|_s^2 \right] \\
&= [\hat{y}_t = y_t] \mathbb{E}_t \left[\left| 1 - \frac{[\hat{y}_t = \tilde{y}_t]}{p_{\tilde{y}_t}} \right|_s^2 \right] \\
&\quad + [\hat{y}_t \neq y_t] \mathbb{E}_t \left[\left| \mathbf{1}_{\hat{y}_t} - \mathbf{1}_{\tilde{y}_t} \frac{[y_t = \tilde{y}_t]}{p_{\tilde{y}_t}} \right|_s^2 \middle| \hat{y}_t \neq y_t \right] \\
&= [\hat{y}_t = y_t] \frac{\gamma(K-1)/K}{1-\gamma+\gamma/K} + \\
&\quad [\hat{y}_t \neq y_t] \left\{ 1 - \frac{\gamma}{K} + \frac{\gamma}{K} \left(1 + \left[\frac{K}{\gamma} \right]^s \right)^{2/s} \right\} \\
&\leq \frac{\gamma}{1-\gamma} + [\hat{y}_t \neq y_t] \left\{ 1 - \gamma + \frac{\gamma}{K} \left(1 + \left[\frac{K}{\gamma} \right]^s \right)^{2/s} \right\}
\end{aligned}$$

□

A.14 Proof of Theorem 11, Chapter 5

We take the expectation of both sides of the equality in (5.8) with respect to \tilde{y}_t , denoted by $\mathbb{E}_t[\cdot]$, and have

$$\begin{aligned}
&\mathbb{E}_t \left[D_{\Phi^*}(U, W^{t-1}) - D_{\Phi^*}(U, W^t) + D_{\Phi^*}(W^{t-1}, W^t) \right] \\
&= \langle W^{t-1} - U, \eta \mathbf{x}_t \tau_t^\top \rangle
\end{aligned}$$

We define $\hat{M}_t = [\hat{y}_t \neq y_t]$. Since $\hat{y}_t \neq y_t$ implies $\nabla \ell_t(W^{t-1}) = \mathbf{x}_t \tau_t^\top$, using the convexity of the loss function, we have

$$\begin{aligned}
(\ell_t(W^{t-1}) - \ell_t(U)) \hat{M}_t &\leq \langle W^{t-1} - U, \nabla \ell_t(W^{t-1}) \rangle \\
&= \langle W^{t-1} - U, \mathbf{x}_t \tau_t^\top \rangle
\end{aligned}$$

We thus have

$$\begin{aligned} & \frac{1}{\eta} \mathbb{E}_t \left[D_{\Phi^*}(U, W^{t-1}) - D_{\Phi^*}(U, W^t) + D_{\Phi^*}(W^{t-1}, W^t) \right] \\ &= \langle W^{t-1} - U, \eta \mathbf{x}_t \tau_t^\top \rangle \geq (\ell_t(W^{t-1}) - \ell_t(U)) \hat{M}_t \end{aligned}$$

By adding the above inequalities of all trials, we have

$$\begin{aligned} & \mathbb{E} \left[\sum_{t=1}^T \ell_t(W^{t-1}) \right] - \left\{ \frac{1}{\eta} D_{\Phi^*}(U) + \sum_{t=1}^T \ell_t(U) \right\} \\ & \leq \frac{1}{\eta} \sum_{t=1}^T \mathbb{E} \left[D_{\Phi^*}(W^{t-1}, W^t) \right] = \sum_{t=1}^T \frac{1}{\eta} \mathbb{E} \left[D_{\Phi}(\theta^{t-1}, \theta^t) \right] \end{aligned}$$

The last step uses the property of Bregman distance in Lemma 6. Since Φ^* is a strictly convex function with constant ρ with respect to $\|\cdot\|_{p,s}$, according to Lemma 6, we have

$$D_{\Phi}(A, B) \leq \frac{1}{2\rho} \|A - B\|_{q,t}^2$$

where $p^{-1} + q^{-1} = 1$ and $s^{-1} + t^{-1} = 1$. Hence,

$$\begin{aligned} \mathbb{E}[D_{\Phi}(\theta^{t-1}, \theta^t)] & \leq \frac{\eta^2}{2\rho} \mathbb{E}[\|\mathbf{x}_t \delta_t^\top\|_{q,t}^2] \\ & \leq \frac{\eta^2}{2\rho} \mathbb{E}[\|\delta_t\|_s^2 \|\mathbf{x}_t\|_p^2] \leq \frac{\eta^2}{2\rho} \mathbb{E}[\|\delta_t\|_s^2] \end{aligned}$$

where the second inequality is due to Holder's inequality. Using the result in Proposition 5 and the fact

$\sum_{t=1}^T \ell_t(W^{t-1}) \hat{M}_t \geq \sum_{t=1}^T \hat{M}_t$, and that $\mathbb{E}[M] \leq \mathbb{E}[\hat{M}] + \gamma T$ we have the result in the theorem.

A.15 Proof of Lemma 7, Chapter 5

$$\begin{aligned}
\langle W - W', \nabla \Phi^*(W) - \nabla \Phi^*(W') \rangle &= \sum_{k=1}^K \sum_{i=1}^d (W_{i,k} - W'_{i,k}) (\ln W_{i,k} - \ln W'_{i,k}) \\
&= \sum_{k=1}^K \sum_{i=1}^d \frac{(W_{i,k} - W'_{i,k})^2}{\bar{W}_{i,k}} \\
&\geq \sum_{k=1}^K \frac{\left(\sum_{i=1}^d |W_{i,k} - W'_{i,k}| \right)^2}{\sum_{i=1}^d \bar{W}_{i,k}}
\end{aligned}$$

The second equality is due to mean value theorem and uses the Taylor expansion of log function where

$\bar{W} = \lambda W + (1 - \lambda)W'$ with $\lambda \in [0, 1]$. Since

$$\frac{\left(\sum_{i=1}^d |W_{i,k} - W'_{i,k}| \right)^2}{\sum_{i=1}^d \bar{W}_{i,k}} \sum_{i=1}^d \bar{W}_{i,k} \geq \left(\sum_{i=1}^d |W_{i,k} - W'_{i,k}| \right)^2$$

and $\sum_{i=1}^d \bar{W}_{i,k} = 1$, we have

$$\langle W - W', \nabla \Phi^*(W) - \nabla \Phi^*(W') \rangle \geq \sum_{k=1}^K \|\mathbf{w}_k - \mathbf{w}'_k\|_1^2$$

Using the property of Bregman distance in Lemma 6 and the fact the dual norm of L_1 is L_∞ , we have the result for $\Phi(\theta)$.

A.16 Proof of Theorem 14, Chapter 6

Proof. Considering that Banditron uses a second-order potential function, we have the following bound when

\hat{y} is used as the predictor:

$$\begin{aligned}
\sum_{t=1}^T \ell_t(W) - \sum_{t=1}^T \ell_t(U) &\leq \|U\|_F^2 + \frac{1}{2} \|W_{t-1} - W_t\|_F^2 \leq \|U\|_F^2 + \frac{1}{2} \mathbb{E} \left[\|\delta_t\|_F^2 \right] \\
&\leq \|U\|_F^2 + \mathbb{E} \left\{ \sum_{t=1}^T \gamma_t [\hat{y}_t = y_t] + \sum_{t=1}^T \frac{K}{\gamma_t} [\hat{y}_t \neq y_t] \right\}
\end{aligned}$$

where we used Theorem 11.1 of [83] and Lemma 6 in the first inequality, $\|\mathbf{x}_t\|_2 \leq 1$ in the second inequality, and Lemma 5 of [5] in the third inequality. Using $E[M] \leq \sum_{t=1}^T \ell_t(W)$ concludes the theorem when we add $E\left[\sum_{t=1}^T \gamma_t\right]$ to get the bound for \tilde{y} [5]. \square

A.17 Proof of Proposition 6, Chapter 6

Proof. We only show the result for $\omega(z) = (a+z)^\lambda$. A similar derivation can be applied to $\omega(z) = \ln(a+z)$.

We have

$$L = \max_{z \geq 0} \frac{\lambda}{(a+z)^{1-\lambda}} = \lambda a^{\lambda-1}$$

To derive ρ , we have

$$\frac{(a+z+t)^{1-\lambda}}{(a+z)^{1-\lambda}} \leq (1+t/a)^{1-\lambda} \leq e^{t(1-\lambda)/a}$$

Hence $\rho = e^{(1-\lambda)/a}$. \square

A.18 Proof of Proposition 7, Chapter 6

Proof. By defining $A_t = \sum_{i=1}^t \tau_i$, we have

$$\sum_{t=1}^T \frac{K}{\gamma_t} \tau_t = \sum_{t=1}^T \frac{2K\omega'_1(A_{t-1})(A_t - A_{t-1})}{\omega'_2\left(\sum_{i=1}^{t-1} 1 + \mu_i\right)} \leq \frac{2K \sum_{t=1}^T \omega'_1(A_{t-1})(A_t - A_{t-1})}{\omega'_2\left(\sum_{i=1}^T 1 + \mu_i\right)}$$

where the inequality is because ω'_2 is a non-increasing function. We also have:

$$\begin{aligned} \sum_{t=1}^T \omega'_1(A_{t-1})(A_t - A_{t-1}) &\leq \rho_1 \sum_{t=1}^T \omega'_1(A_t)(A_t - A_{t-1}) \\ &\leq \rho_1 \sum_{t=1}^T (\omega_1(A_t) - \omega_2(A_{t-1})) \leq \rho_1 \omega_1(A_T) \end{aligned} \quad (\text{A.7})$$

where the first step is due to the definition of good support functions, the second step is due to the concavity of ω_1 and the last step is due to the telescope property and the fact that $\omega_1(0) \geq 0$. Combining the above

results produces the first inequality in the proposition. The proof of the second inequality in the proposition is similar and follows. By defining $B_t = \sum_{i=1}^t \mu_i$, we have

$$\begin{aligned} \sum_{t=1}^T \gamma_t \mu_t &= \sum_{t=1}^T \frac{\omega'_2(B_{t-1} + t - 1)}{2\omega'_1(A_{t-1})} (B_t - B_{t-1}) \leq \sum_{t=1}^T \frac{\omega'_2(B_{t-1})}{2\omega'_1(A_{t-1})} (B_t - B_{t-1}) \\ &\leq \frac{\sum_{t=1}^T \omega'_2(B_{t-1})(B_t - B_{t-1})}{2\omega'_1(A_T)} \leq \frac{\sum_{t=1}^T \rho_2^2 \omega'_2(B_t)(B_t - B_{t-1})}{2\omega'_1(A_T)} \leq \frac{\rho_2^2 \omega_2(B_T)}{2\omega'_1(A_T)} \end{aligned}$$

Notice the third equality is because $\omega'_2(B_{t-1}) \leq \rho_2^{\mu_t} \omega'_2(B_t)$ and $\mu_t \leq 2$. Similarly, we have:

$$\begin{aligned} \sum_{t=1}^T \gamma_t &= \sum_{t=1}^T \frac{\omega'_2(B_{t-1} + t - 1)}{2\omega'_1(A_{t-1})} [t - (t - 1)] \leq \sum_{t=1}^T \frac{\omega'_2(t - 1)}{2\omega'_1(A_{t-1})} [t - (t - 1)] \\ &\leq \frac{1}{2\omega'_1(A_T)} \sum_{t=1}^T \omega'_2(t - 1) [t - (t - 1)] \leq \frac{\rho_2}{2\omega'_1(A_T)} \sum_{t=1}^T \omega'_2(t) [t - (t - 1)] \\ &\leq \frac{\rho_2 \omega_2(T)}{2\omega'_1(A_T)} \end{aligned}$$

□

A.19 Proof of Proposition 8, Chapter 6

Proof. Similar to the argument in Section 6.3.2, we have

$$\sum_{t=1}^T \gamma_t \mu_t = \sum_{t=1}^T \frac{1}{2L} \omega' \left(\sum_{i=1}^{t-1} \mu_i \right) \mu_t = \sum_{t=1}^T \frac{1}{2L} \omega' (A_{t-1}) (A_t - A_{t-1}) \leq \frac{\rho^2}{2L} \omega(A_t)$$

A similar argument can be applied to prove the second and third inequality in the proposition. □

BIBLIOGRAPHY

Bibliography

- [1] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge Univ Pr, 2006.
- [2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.
- [3] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.
- [4] Robert E. Schapire. The strength of weak learnability. *Journal of Machine Learning*, 5:197–227, 1990.
- [5] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *International Conference on Machine Learning '08*, pages 440–447, 2008.
- [6] Joannès Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European Conference on Machine Learning*, pages 437–448. Springer, 2005.
- [7] Alon Altman and Moshe Tennenholtz. Ranking systems: the pagerank axioms. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 1–8, New York, NY, USA, 2005. ACM.
- [8] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.
- [9] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.
- [10] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. Learning to rank by optimizing ndcg measure. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1883–1891, 2009.
- [11] T. Joachims. Optimizing search engines using clickthrough data. In *SIGKDD*, pages 133–142, 2002.
- [12] Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Now Publishers Inc, 2009.
- [13] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR 2007*, pages 271–278, 2007.

- [14] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR 2000*, pages 41–48, 2000.
- [15] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2001.
- [16] Ping Li, Christopher Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Neural Information Processing Systems 2007*, Cambridge, MA, 2008.
- [17] Ramesh Nallapati. Discriminative models for information retrieval. In *SIGIR 2004*, pages 64–71, 2004.
- [18] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support vector learning for ordinal regression. In *ICANN 1999*, pages 97–102, 1999.
- [19] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.
- [20] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning*, pages 89–96, 2005.
- [21] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *SIGIR 2006*, pages 186–193, 2006.
- [22] Ming Feng Tsai, Tie yan Liu, Tao Qin, Hsin hsi Chen, and Wei ying Ma. Frank: A ranking method with fidelity loss. In *SIGIR 2007*, 2007.
- [23] Rong Jin, Hamed Valizadegan, and Hang Li. Ranking refinement and its application to information retrieval. In *WWW 2008*, pages 397–406, 2008.
- [24] Tao Qin, Tie yan Liu, Ming feng Tsai, Xu dong Zhang, and Hang Li. Learning to search web pages with query-level loss functions. Technical report, 2006.
- [25] Christopher J. C. Burges, Robert Ragno, and Quoc V. Le. Learning to rank with nonsmooth cost functions. In *Neural Information Processing Systems 2006*, 2006.
- [26] Zhe Cao and Tie yan Liu. Learning to rank: From pairwise approach to listwise approach. In *International Conference on Machine Learning 2007*, pages 129–136, 2007.
- [27] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *International Conference on Machine Learning 2008*, pages 1192–1199, 2008.
- [28] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *WSDM 2008*, pages 77–86, 2008.
- [29] Maksims Volkovs and Richard S. Zemel. Boltzrank: learning to maximize expected ranking gain. In *International Conference on Machine Learning*, page 137, 2009.
- [30] K. P. Bennett and Ayhan Demiriz. Semi-supervised support vector machine. In *Neural Information Processing Systems*, 1999.

- [31] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *10th Int. Workshop on AI and Stat*, 2005.
- [32] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning*, 2001.
- [33] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, 2003.
- [34] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Neural Information Processing Systems*, 2003.
- [35] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semisupervised learning on large graphs. In *International Conference on Learning Theory*, 2004.
- [36] Yoav Freund. Boosting a weak learning algorithm by majority. In *COLT '90: Proceedings of the third annual workshop on Computational learning theory*, pages 202–216, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.
- [37] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999.
- [38] Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. On the convergence of bound optimization algorithms. In *Uncertainty in Artificial Intelligent*, pages 509–516, 2003.
- [39] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, 1996.
- [40] Pavan Kumar Mallapragada, Rong Jin, Anil K. Jain, and Yi Liu. Semiboost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:2000–2014, 2009.
- [41] Yi Liu, Rong Jin, and Anil K. Jain. Boostcluster: boosting clustering by pairwise constraints. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 450–459, New York, NY, USA, 2007. ACM.
- [42] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR 2007*, pages 391–398, 2007.
- [43] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65:386–408, 1958.
- [44] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal of Computing*, 32(1):48–77, 2003.
- [45] C. Watkins. *Learning from delayed Rewards*. PhD thesis, Cambridge, 1989.
- [46] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Learning Theory '02*, pages 255–270, 2002.
- [47] Shie Mannor and John N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004.

- [48] K. P. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *KDD*, 2002.
- [49] K. Chen and S. Wang. Regularized boost for semi-supervised learning. In *Neural Information Processing Systems*, 2007.
- [50] Hamed Valizadegan, Rong Jin, and Anil K. Jain. Semi-supervised boosting for multi-class classification. In *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, pages 522–537, Berlin, Heidelberg, 2008. Springer-Verlag.
- [51] S. Robertson and D. A. Hull. The trec-9 filtering track final report. In *TREC9*, pages 25–40, 2000.
- [52] Koby Crammer, Yoram Singer, and K. Warmuth. Ultraconservative online algorithms for multiclass problems. 3:2003, 2003.
- [53] Shijun Wang, Rong Jin, and Hamed Valizadegan. A potential-based framework for online multi-class learning with partial feedback. In *International Conference on Artificial Intelligence and Statistics '10*, 2010.
- [54] A. Frank and A. Asuncion. UCI machine learning repository, 2010.
- [55] Chih-Chung Chang and Chih-Jen Lin. Libsvm : a library for support vector machines, 2001.
- [56] Tie-Yan Liu, Tao Qin, Jun Xu, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR 2007*, 2007.
- [57] GroupLens. *MovieLens Data sets*. <http://www.grouplens.org/node/12>, 2006.
- [58] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. AI Res.*, 2:263–286, 1995.
- [59] Rong Jin and Jian Zhang. Multi-class learning by smoothed boosting. *Mach. Learn.*, 67(3):207–227, 2007.
- [60] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.
- [61] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*, 2002.
- [62] Günther Eibl and Karl-Peter Pfeiffer. Multiclass boosting for weak classifiers. *J. Mach. Learn. Res.*, 6:189–210, 2005.
- [63] Ling Li. Multiclass boosting with repartitioning. In *International Conference on Machine Learning*, 2006.
- [64] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Science, University of Wisconsin-Madison, 2005.
- [65] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

- [66] T. De Bie and N. Cristianini. Convex methods for transduction. In *Neural Information Processing Systems*, 2004.
- [67] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *AAAI*, 2005.
- [68] F. d’Alche Buc, Y. Grandvalet, and C. Ambroise. Semi-supervised marginboost. In *Neural Information Processing Systems*, 2002.
- [69] Nicholas J. Higham. Matrix nearness problems and applications. In M. J. C. Gover and S. Barnett, editors, *Applications of Matrix Theory*, pages 1–27. Oxford University Press, 1989.
- [70] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- [71] Jen-Yuan Yeh, Yung-Yi Lin, Hao-Ren Ke, and Wei-Pang Yang. Learning to rank for information retrieval using genetic programming. In *LR4IR 2007*, 2007.
- [72] Zhengya Sun, Tao Qin, Qing Tao, and Jue Wang. Robust sparse rank learning for non-smooth ranking measures. In *SIGIR*, pages 259–266, 2009.
- [73] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, pages 111–119, 2001.
- [74] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis. In *Uncertainty in Artificial Intelligent*, 2000.
- [75] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, 546 pp, 1986.
- [76] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132, 2000.
- [77] J. Gao, H. Qi, X. Xia, and J.-Y. Nie. Discriminant model for information retrieval. In *SIGIR*, pages 290–297, 2005.
- [78] D. Harman. Relevance feedback revisited. In *SIGIR*, 1992.
- [79] Mark Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. In *CIKM ’02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 538–548. ACM, 2002.
- [80] R.T. Rockafellar. *Convex analysis*. Princeton University Press, Princeton, N.J., 1970.
- [81] Robert E. Schapire. Theoretical views of boosting and applications. In *Algorithmic Learning Theory, 10th International Conference, ALT ’99*, volume 1720, pages 13–25. Springer, 1999.
- [82] J. J. Rocchio. Relevance feedback in information retrieval. 1971.
- [83] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

- [84] Donald A. Berry and Bert Fristedt. *Bandit problems: Sequential allocation of experiments*. Chapman and Hall, 1985.
- [85] John Langford and Zhang Tong. The epoch-greedy algorithm for contextual multi-armed bandits. In *Neural Information Processing Systems '07*, 2007.
- [86] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Journal of Machine Learning*, 2(4):285–318, April 1988.
- [87] Jyrki Kivinen and Manfred K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *ACM Symposium on Theory of Computing '95*, pages 209–218, 1995.
- [88] Adam J. Grove, Nick Littlestone, and Dale Schuurmans. General convergence results for linear discriminant updates. *Journal of Machine Learning*, 43(3):173–210, 2001.
- [89] Herbert Robbins. some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952.
- [90] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.
- [91] Chih chun Wang, Student Member, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50:338–355, 2005.
- [92] Vijaykumar Gullapalli Computer and Vijaykumar Gullapalli. Associative reinforcement learning of real-valued functions. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, 1991.
- [93] Alexander L. Strehl, Chris Mesterharm, Michael L. Littman, and Haym Hirsh. Experience-efficient learning in associative bandit problems. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 889–896, New York, NY, USA, 2006. ACM.
- [94] Chih chun Wang, Sanjeev R. Kulkarni, and H. Vincent Poor. Arbitrary side observations in bandit problems. *Adv. Applied Math*, 34:903–936, 2005.
- [95] Nicolas Meuleau and Paul Bourgin. Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Mach. Learn.*, 35(2):117–154, 1999.
- [96] Nicolò Cesa-Bianchi and Paul Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *In 5th International Conference on Machine Learning*, pages 100–108. Morgan Kaufmann, 1998.
- [97] Kolby Crammer. Online learning of real-world problems. In *International Conference on Machine Learning '07*, 2007.
- [98] Iead Rezek, David S. Leslie, Steven Reece, Stephen J. Roberts, Alex Rogers, Rajdeep K. Dash, and Nicholas R. Jennings. On similarities between inference in game theory and machine learning. *J. Artif. Intell. Res. (JAIR)*, 33:259–283, 2008.
- [99] Amy R. Greenwald and Michael L. Littman. Introduction to the special issue on learning and computational game theory. *Machine Learning*, 67(1-2):3–6, 2007.

MICHIGAN STATE UNIVERSITY LIBRARIES



3 1293 03220 8740