NEW LEARNING FRAMEWORKS FOR INFORMATION RETRIEVAL

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yisong Yue

January 2011

NEW LEARNING FRAMEWORKS FOR INFORMATION RETRIEVAL

Yisong Yue, Ph.D.

Cornell University 2011

Recent advances in machine learning have enabled the training of increasingly complex information retrieval models. This dissertation proposes principled approaches to formalize the learning problems for information retrieval, with an eye towards developing a unified learning framework. This will conceptually simplify the overall development process, making it easier to reason about higher level goals and properties of the retrieval system. This dissertation advocates two complementary approaches, structured prediction and interactive learning, to learn feature-rich retrieval models that can perform well in practice.

ACKNOWLEDGEMENTS

First and foremost, I am deeply indebted to my Ph.D. advisor, Thorsten Joachims, for his invaluable and inspirational insight, as well as endless kindness and encouragement. This dissertation would not have been possible without the guidance, wisdom and support he provided during my time at Cornell. I am also indebted to Bobby Kleinberg, who I hope instilled in me some of his abilities to give amazingly clear explanations and maintain impeccable technical rigor.

The Department of Computer Science at Cornell truly offers a world-class research and learning environment. I was continuously amazed by the generosity, brilliance and energy of the faculty, many of whom aided me in numerous ways during my time there. I especially would like to thank Ramin Zabih, Jon Kleinberg and Rich Caruana for teaching such fantastic courses and stimulating the imagination of a young first-year Ph.D. student.

During my graduate years, I managed to get away to Microsoft, Google and Yahoo! for very valuable and eye-opening internships and extended visits. I want to thank my mentors and collaborators, Chris Burges, Rajan Patel and Olivier Chapelle, for working with me and showing me refreshing new perspectives. I am particularly grateful to Chris, who provided a significant amount of mentorship during a formative time in my grad school career. I very much enjoyed having Friday tea with Chris's research group, during which our fearless leader would force his minions to derive formulations and proofs in between sipping on Earl Grey with a half a cup of milk and one cube of sugar.

I thank my Ph.D. committee, Thorsten Joachims, Bobby Kleinberg, Chris Burges, Ping Li and John Hopcroft, for their helpful comments and suggestions.

I am also grateful to my fellow machine learning colleagues at Cornell; we grew together into researchers. Special thanks to Filip Radlinski and Tom Finley, who

helped guide me on my first research project. Also thanks to Nikos Karampatziakis and Art Munson for organizing the machine learning discussion group with me. It was a great venue for brainstorming, debating and socializing. And, of course, thanks to Nikos and Ainur Yessenalina for being my office neighbors and listening to all of my half-baked ideas.

I want to give a shout-out to my other fellow Cornellians, who made life fun and interesting in many ways during my grad school years. From singing with the Chorus & Glee Club (especially ACDA and China Tour), movie nights at the Grindhouse, playing Fish Bowl at the Dollhouse, TGIF at the Big Red Barn, game nights, picnics, playing ice hockey, holiday parties, morning wine at CTB, and even adventures at Wegmans or during driving lessons, there was nary a dull moment up on the Hill. Special thanks to Aaron, Allison, Caroline, Fang, Kristel and Ryan. Thanks for the memories.

As always, I am grateful to my everlasting high school friends – especially Dan, John, Navreet and Steve – who have continuously pushed and motivated me mentally in the twelve years we have known each other.

Finally and most humbly, I thank my family for their incredible love, sacrifice and support. It is first through their care and perseverance that I was gifted with such amazing opportunities. Oh, and thanks to my sister, Lena, for doing my laundry.

This research was funded by the National Science Foundation under Grants IIS-0713483, IIS-0812091, and IIS-0905467, by a Microsoft Research Graduate Fellowship and a Yahoo! Key Scientific Challenges Award, and by travel support from the Association for Computing Machinery Special Interest Group on Information Retrieval, Microsoft Research, and the International Machine Learning Society.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part I

Overview and Preliminaries

CHAPTER 1

INTRODUCTION

This dissertation describes advances towards a more unified framework for learning information retrieval models. Recent progress in machine learning has strongly impacted the development of state-of-the-art retrieval systems. For instance, online search, which is a multi-billion dollar industry, relies heavily on machine learning when designing retrieval models. But while commercial search engines have been very successful, it remains unclear how we can systematically improve performance and model more complex retrieval paradigms. Having a unified framework supported by theory simplifies the task of analyzing complex modeling problems, thus making it much easier to reason about higher level goals and properties of the retrieval system.

The research described herein is motivated by the following general observations: (1) the utility that users derive from using an information retrieval system is a highly complex function that depends on several interacting factors, (2) information retrieval systems do not exist in a vacuum, but rather must interact with users. Properly understanding and leveraging the associated underlying technical issues can help us more effectively model user utility jointly over multiple factors as well as automatically tune system parameters via intelligently interacting with users and collecting feedback.

Building upon these motivations, this dissertation describes two complementary approaches for designing expressive and robust models with effective training and prediction methods. First, information retrieval can be formulated as a structured prediction problem, which can jointly model the interdependencies between the predictions (e.g., documents in a ranking). Second, optimizing an information

retrieval system using user feedback can be formulated as an interactive learning problem, where the system adaptively chooses how to respond to information requests in order to simultaneously both provide good service and also learn from user feedback. A related important problem is proper interpretation of user feedback: accurately interpreting implicit user feedback[1] is vital to the design of interactive systems, since otherwise we might be optimizing an incorrect objective. For the remainder of this chapter, we briefly introduce these aforementioned research directions and the corresponding contributions of this dissertation.

1.1 Structured Prediction

In order to effectively respond to a wide range of information needs, a retrieval model must be rich enough to capture the essential qualities which discriminates between good and poor results. Another important component is the objective function to be optimized during model-parameter tuning (i.e. learning). This objective function must be expressive enough to accurately capture user intent. Both of these goals can be formulated as structured prediction problems.

Broadly speaking, structured prediction refers to any type of prediction performed jointly over multiple input instances (e.g., a ranking over a list of documents). Rankings are the most common types of structured outputs within information retrieval. Indeed, structured prediction is not a new idea. But until recently, the primary impediment has been a lack of efficient and robust methods for training. One can naively incorporate all possible information and constraints in solving an optimization problem, but this approach rather quickly becomes too expensive and also does not generalize well to unseen test data. Most previous

[1]Implicit feedback makes up the majority of the feedback that can be derived from observing users' interactions.

work focused instead on modeling individual documents and their relevance to individual queries and information needs.

Within this context, the contributions of this dissertation are two-fold. First, we propose a novel framework for optimizing rank-based performance measures commonly used to evaluate retrieval methods. The complexity of these measures make direct optimization difficult when using conventional machine learning methods. This dissertation proposes the first structured prediction learning algorithm that provably optimizes a rigorous upper bound on the rank-based performance measure average precision (and can be extended to other measures as well).

Second, this dissertation proposes a structured prediction approach for diversified retrieval. Diversified retrieval has recently become a popular research topic, as many information retrieval researchers have noted the need to reduce redundancy and to deal with ambiguity when responding to information requests. In this dissertation, I will show that interdependencies such as redundancy of information between documents can be naturally modeled using structured prediction approaches.

These approaches are described in greater detail in Chapters 3 and 4, which include a description of the model, training and prediction algorithms, theoretical guarantees, and empirical evaluations.

1.2 Interactive Learning

One potential limitation of batch learning algorithms (which includes many learning algorithms for structured prediction) is the assumption that the training data

is representative of unseen test instances. Current approaches to learning and evaluating retrieval models are largely restricted to supervised learning techniques using large amounts of expensive training data labeled by a small number of human judges. Due to this substantial cost, such datasets are typically not fully representative of the natural usage contexts of real search engines.

The key observation here is that information retrieval systems do not exist in a vacuum; they must interact with users. The information we search for, the web pages we browse, the emails we send, the Twitter tweets we post, the items we purchase on Amazon – they all leave digital footprints that reflect the fine grained dynamics of our online activities. These interactions are plentiful and can be harvested at virtually no cost. This naturally beckons for developing systems that can adaptively reconfigure or tune themselves in new environments by intelligently interacting with users.

This dissertation proposes a novel online learning framework, called the Dueling Bandits Problem, tailored towards real-time learning from user behavior in information retrieval systems. In particular, this framework only requires pairwise comparisons, which were shown to be reliably inferred in search applications. Chapters 5 and 6 describe this framework in greater detail, and include provably efficient algorithms for optimizing over both discrete and continuous hypothesis classes of retrieval functions. Furthermore, the proposed algorithms are simple, making them easy to implement and extend in a variety of applications.

1.3 Interpreting User Feedback

Machine learning algorithms typically assume the availability of explicit feedback (e.g., human annotated training labels). However, the overwhelming majority of feedback collected from users' interactions with an information retrieval system will be implicit in nature. The final contribution of this dissertation is a collection of methods that can learn to better interpret implicit user feedback.

Clicks are the most plentiful form of user interactions. But a clicked result need not be a good result. As a simple thought experiment, consider two rankings, where one has good results and the other mediocre results. Which ranking would you expect to receive more clicks? On one hand, all the results in the first ranking are good. But users might click around more in the second ranking to collect more information. Such questions must be resolved in order for computers systems to tease out useful information from observed behavior. This dissertation proposes methods for making the data collection process more efficient by learning more informative interpretations of user feedback; the methods and experiments are described in Chapter 7.

CHAPTER 2

LEARNING TO RANK: A BRIEF HISTORY

Information retrieval (IR) has been an active research area since the 1960s. Popular techniques used today include the vector space model [152], TF-IDF [150], Okapi BM25 [144], and language modeling approaches [131, 107, 188, 170]. While some approaches offer deeper theoretical discussions than others, in practice, virtually all successful methods reduce to using a similarity function to compute the compatibility (or relevance) of a document to a query [121]. Assuming that relevance of documents are independent, then the optimal ranking (to present to users) results from sorting by the compatibility scores (this is known as the Probability Ranking Principle [143]).

Traditional IR methods typically employ relatively simple similarity functions with few or no parameters. As such, they are easy to tune and have been shown to generalize well to new corpora. However, over the past fifteen years, we have seen a trend towards using richer models which utilize large feature spaces. As a result, one now must automatically find good models from large parameter spaces. Due to its practical importance as well as natural formulation as a learning problem, both the machine learning (ML) and information retrieval (IR) communities have accordingly shown growing interest in the problem of learning ranking functions (see [117] for a longer survey of Learning to Rank).

Let $\mathcal{D}$ and $\mathcal{Q}$ be the space of documents and queries, respectively. In general, the basic learning goal is to find a scoring function $h : \mathcal{Q} \times \mathcal{D} \rightarrow \mathcal{R}$ such that, for a given query q and set of documents $\{d_1, \ldots, d_{n_q}\}$, the induced ranking from sorting by $h(q, d_i)$ is "good" under certain criteria. The similarity between documents and

queries can be captured using an appropriate feature map ϕ,

$$x \equiv \phi(q, d) \subseteq R^m, \tag{2.1}$$

which maps queries and documents to a high dimensional feature space. Following standard ML notation, these input instances are typically denoted as x, and the space of input instances is denoted as $\mathcal{X}$. Discovering useful features for learning is an area of ongoing research.

We can thus state our goal as learning a hypothesis function $h : \mathcal{X} \rightarrow \mathcal{Y}$ between an input space $\mathcal{X}$ and output space $\mathcal{Y}$ (e.g., $\mathcal{Y}$ could be a space of rankings). In order to quantify the quality of a prediction, $\hat{y} = h(x)$, we will consider a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re$. $\Delta(y, \hat{y})$ quantifies the penalty for making prediction $\hat{y}$ if the correct output is y. In the supervised learning scenario,[1] where input/output pairs (x, y) are available for training and are assumed to come from some fixed distribution $P(x, y)$, the goal is to find a function h such that the risk (i.e., expected loss),

$$R_P^{\Delta}(h) = \int_{\mathcal{X} \times \mathcal{Y}} \Delta(y, h(x)) dP(x, y),$$

is minimized. Of course, $P(\mathbf{x}, \mathbf{y})$ is unknown. But given a finite set of training pairs, $S = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, N\}$, the performance of h on S can be measured by the empirical risk,

$$R_S^{\Delta}(h) = \frac{1}{N} \sum_{i=1}^{N} \Delta(y_i, h(x_i)).$$

For simplicity and ease of notation, we will typically restrict our discussion to linear hypothesis functions,

$$h(x|w) = w^T x, \tag{2.2}$$

[1]Supervised learning is the most common learning setting. The drawback of requiring labeled data (which can be expensive to obtain) is a major motivation for the interactive learning approach discussed in Chapters 5 and 6.

although other function classes can be used. In this setting, the learning goal reduces to finding a model vector w which optimizes an appropriate objective function. The challenge lies in choosing objective functions which not only accurately measure the quality of the induced rankings, but also yield efficiently computable solutions.

2.1 Classification & Regression

One can treat learning ranking functions as a conventional classification or regression problem. In the classification setting, one assumes that training inputs are categorized into discrete classes, and the learned hypothesis function discriminates between instances of different classes. In the simple case of binary classification (two classes), most approaches learn a decision threshold b along with a model vector w such that predictions are made via $sign(w^T x - b)$. The specific objective function formulation varies depending on assumptions regarding the learning problem, but the high level goal is maximizing accuracy. For example, the popular SVM training algorithm uses the principle of structural risk minimization [167, 153] to learn a model w that separates the two classes by as large a margin as possible. Let y_i denote the label of the ith training instance, with relevant and non-relevant labels taking on values of +1 and -1, respectively. The SVM formulation can be written as OP 1.

Optimization Problem 1. (SVM)

$$\underset{w,b,\xi \geq 0}{\operatorname{argmin}} \ \frac{1}{2}\left(\|w\|^2 + b^2\right) + \frac{C}{N}\sum_{i=1}^{m}\xi_i \tag{2.3}$$

$$s.t.\ \forall i \in \{1,\ldots,N\}:$$

$$y_i(w^T x_i - b) \geq 1 - \xi_i \tag{2.4}$$

Each input example has a corresponding slack variable ξ_i which penalizes margin violations. Note that $\xi_i \geq 1$ if training instance x_i is misclassified. Thus we can see that the sum of slacks $\sum \xi_i$ defines a smooth upper bound on accuracy loss. Other popular classification methods include boosting [67], neural nets [124, 17, 18], decision trees [21, 124, 18], perceptrons [147, 70, 68], and naïve Bayesian networks [57, 124, 18].

In the regression setting, one aims to learn models whose output scores match the target labels. For linear regression, this amounts to learning a w and a bias b to produce output scores of the form $w^T x + b$. The most popular objective minimizes the sum of squared error, $\sum (y_i - (w^T x_i + b))^2$, which can be justified from assuming a probabilistic model that has i.i.d. Gaussian noise on the target labels. Since squared error is differentiable everywhere, gradient descent techniques can easily find a local optimum. For linear regression, the objective function is also convex and has a closed form solution. Other popular regression methods include neural nets [124, 17, 18], decision trees [21, 124, 18], Gaussian processes [141], and logistic regression [124, 18].

While classification and regression approaches have proven effective for many IR tasks, they optimize for loss functions which can conflict with the goal of predicting good rankings. First, the relevance labels induce a weak ordering on the input examples. Multi-class classification approaches ignore this ordinal structure completely and assume the class labels do not interact. On the other hand, regression approaches consider more restrictive goals by requiring output scores to match their target labels. Second, many datasets exhibit a large class imbalance, where most of the input instances are not relevant. Consider an extreme case where 99.9% of the instances are non-relevant. Then a baseline hypothesis which always

predicts 0 will achieve an average accuracy of 0.999. As such, models which achieve the highest accuracy might not produce the best possible rankings. Subsequent sections describe approaches that explicitly learn to predict good rankings.

2.2 Ordinal Regression

Ordinal regression refers to prediction problems where the target labels have an ordinal structure. The goal then is to predict output scores whose induced ranking agrees with the weak ordering defined by the ordinal class labels. Let the ordinal classes be $\{0, 1, \ldots, T\}$. The learned hypothesis function must score instances labeled 1 higher than those labeled 0, those labeled 2 higher than those labeled 1, and so forth. In contrast to multiclass prediction, ordinal regression explicitly penalizes disgreement with the label ordering. In contrast to conventional regression, ordinal regression does not require that the output scores match the class label values.

2.2.1 Learning Multiple Thresholds

One natural approach to ordinal regression involves learning thresholds which separates instances of different classes. The goal can be expressed as learning thresholds $b_1 < b_2 < \ldots < b_T$ along with a model vector w such that, for instances with ordinal label j, we have $b_j < w^T x < b_{j+1}$. This general principle can be adapted to many different learning algorithms. For example, using SVMs [42], one can find the w that maximizes the margin of the output scores $w^T x$ from their corresponding thresholds. This leads to an extension of conventional SVMs which uses multiple

decision thresholds (as opposed to just one b), as well as two slack variables for each input instance (for both upper and lower threshold margin violation). The SVM formulation can be written as OP 2. Other prior work have also adapted this approach to Gaussian processes [41], decision trees [102], perceptrons [49], and neural nets [33].

Optimization Problem 2. (Multiple Threshold Ordinal SVM)

$$\underset{w,\mathbf{b},\xi\geq 0}{\operatorname{argmin}} \ \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{j=1}^{T}\left(\sum_i \xi_{i,j}^{+} + \sum_i \xi_{i,j}^{-}\right) \tag{2.5}$$

$$s.t.\ \forall j \in \{0,\ldots,T\}:$$

$$w^T x_i - b_j \geq 1 - \xi_{i,j}^{-}, \quad \forall i \ : \ y_i = j \tag{2.6}$$

$$w^T x_i - b_j \geq -1 + \xi_{i,j}^{+}, \quad \forall i \ : \ y_i = j-1 \tag{2.7}$$

$$b_1 < b_2 < \ldots < b_T \tag{2.8}$$

2.2.2 Decomposition into Multiple Training Sets

Another approach, which has seen increased interest in applications to IR, decomposes ordinal regression into smaller classification or regression problems. The most common approach is to use T classifiers $w_1,\ldots,w_T$, where w_i learns to discriminate between classes $\{0,\ldots,i-1\}$ and $\{i,\ldots,T\}$. The final output score is then a combination of the outputs of all T classifiers (e.g., the sum). Existing methods adopting this approach include using SVMs [132], as well as using boosted decision trees [113]. These two methods were specifically designed for learning retrieval models, and are also applicable beyond ordinal regression since the decomposition into smaller classification tasks can be motivated by other types of label structure. More general decomposition techniques also exist which can convert any type of

multi-class classification problem (of which ordinal regression is a special case) into a collection of binary classification problems [16].

2.2.3 Optimizing Pairwise Preferences

The most popular ordinal regression approach used for IR decomposes the learning problem into pairwise preferences. Any pair of instances (x, y) and (x', y'), with $y > y'$, generates a preference that the learned model should satisfy $w^T x > w^T x'$. In the case where relevance is binary (relevant or not relevant), the fraction of satisfied pairwise preferences is equivalent to the ROC-Area measure.

Optimizing over pairwise preferences reduces to generating a dataset (often represented implicitly) where each input instance has the form $(x - x')$ with label $+1$. Correctly predicting $w^T(x - x') > 0$ indicates that the preference $w^T x > w^T x'$ is satisfied. Optimizing pairwise preferences can be applied to almost any learning method. For example, an SVM approach [75, 85, 29] might require that the preferences be satisfied by as large a margin as possible. The SVM formulation can be written as OP 3.

Optimization Problem 3. (PAIRWISE SVM)

$$\underset{w,\xi \geq 0}{\operatorname{argmin}} \ \frac{1}{2}\|w\|^2 + \frac{C}{\#pairs} \sum_{(i,j):y_i > y_j} \xi_{i,j} \tag{2.9}$$

$$s.t.\ \forall i, j \ \ where\ y_i > y_j \ \ :$$

$$w^T(x_i - x_j) \geq 1 - \xi_{i,j} \tag{2.10}$$

Other work on optimizing pairwise preferences include using neural nets [26], logistic regression [76, 32], and boosting [66, 44, 119], and more general model-

agnostic approaches [14, 5]. Optimizing pairwise preferences also naturally deals with severe class imbalances, which is easy to see in the binary case due to its equivalence to optimizing ROC-Area. Like the methods described in Section 2.2.2, many methods mentioned in this section were designed specifically for learning retrieval models [26, 32, 29], and have been shown to perform well empirically. These methods can also be applied beyond ordinal regression since pairwise preferences can be generated in other ways.

2.3 Rank-based Performance Measures

The different approaches described in Section 2.2 can be thought of as optimizing an objective function defined over rankings. The popular pairwise approaches described in Section 2.2.3 use objective functions which decomposes into a sum over pairwise preference agreements. As stated previously, optimizing pairwise preferences over binary relevance labels is equivalent to optimizing ROC-Area, which is a well-known rank-based performance measure.

ROC-Area weighs all positions in the ranking equally. In contrast, studies have shown that users typically focus on the very top of presented rankings [71, 88]. Likewise, the IR community primarily uses rank-based measures which emphasize the top of the ranking [82, 145]. Among the most common measures are precision@k, mean average precision, mean reciprocal rank, and normalized discounted cumulative gain.

Precision@k - precision@k is a binary relevance measure and refers to the percentage of relevant documents amongst the top k documents. For example, the

ranking

$$0\ 1\ 0\ 1\ 0 \tag{2.11}$$

has precision@1 = 0, precision@2 = 1/2, precision@3 = 1/3, and precision@4 = 2/4. Typical values of k for IR studies range between 1 and 10. The special case of $k = 1$ is also known as winner takes all. This measure emphasizes the top of rankings by simply not considering any documents lower than rank k.

Mean Average Precision (MAP) - average precision is a binary relevance measure that is computed by averaging the precision@k scores for k equal to the rank position of each relevant document. MAP is then the mean of average precision scores over a group of queries. For example, the ranking in (2.11) has relevant documents in the second and fourth rank positions, thus yielding precision@2 = 0.5 and precision@4 = 0.5. The average precision of the above ranking is then $(0.5 + 0.5)/2 = 0.5$. In contrast, the ranking

$$1\ 0\ 0\ 0\ 1 \tag{2.12}$$

has relevant documents in the first and fifth positions, yielding precision@1 = 1 and precision@5 = 0.4. The average precision of the above ranking is then $(1 + 0.4)/2 = 0.7$. Changing from (2.11) to (2.12) requires moving one relevant document up a rank and one down a rank. Since greater emphasis is placed at the top of the ranking, average precision improves from (2.11) to (2.12).

Mean Reciprocal Rank (MRR) - reciprocal rank is a binary relevance measure that results from computing the reciprocal rank value of the highest ranked relevant document. MRR is then the mean of reciprocal rank scores over a group of queries. For example, the ranking in (2.11) has a reciprocal rank of 1/2, whereas the ranking in (2.12) has a reciprocal rank of 1.

Normalized Discounted Cumulative Gain (NDCG) - NDCG is scored for documents with multiple levels of relevance. Discounted cumulative gain (DCG) can be computed as

$$DCG@k = \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log(i+1)},$$

where $r_i \in \{0, \ldots, T\}$ is the relevance level of the document a rank i. One can think of DCG@k as a sum of document scores up to rank k, where more relevant documents are given exponentially larger scores, and all scores are subject to logarithmic decay with respect to rank position. NDCG@k is then a normalized version of DCG@k such that the best possible ranking has NDCG@$k = 1$. When the measure is computed over the entire ranking (instead of up to a rank limit k), it is refered to simply as NDCG.

Explicitly optimizing for rank-based measures is the subject of very recent and ongoing research within the ML and IR communities. Initial approaches to optimizing rank-based measures either used ad-hoc heuristics [29], or optimized over very restricted parameter spaces [123]. Chapter 3 describes current approaches (including contributions of this dissertation) to optimizing rank-based measures by viewing it as a structured prediction problem.

2.4 Diversified Retrieval: Beyond Independent Relevance

Standard retrieval functions typically consider the relevance of each document independently of other documents. While this greatly reduces the modeling complexity and yields efficient prediction algorithms (i.e., sorting by the output scores), it limits the ability of such retrieval functions to model more complex criteria. One rapidly growing research direction in IR is *diversified retrieval*, which is concerned

with suppressing information redundancy, or encouraging diversity and novelty, in the retrieved results. The performance measures described in Section 2.3 do not penalize redundancy. Indeed, it is unclear (and will likely vary depending on the retrieval task and domain) how well measures such as NDCG actually correlate with real user satisfaction [6]. Recent studies have noted the necessity of modeling inter-document dependencies when explicitly optimizing diversity in IR [31, 186, 40, 156, 189, 43, 92]. Some studies have also proposed new performance measures which penalize redundancy in the retrieved results [186, 43].

Two particular settings which clearly benefit from improved diversity are ambiguous queries [90] and learning or informational queries [156]. Ambiguous queries arise from users with different information needs issuing the same textual query. For example, the query "Jaguar" can refer to many different topics such as the car or the feline. It might be wise to retrieve at least one relevant result for each topic. On the other hand, users issuing learning queries are interested in "a specific detail or the entire breadth of knowledge available" for a specific query [156]. Thus, the retrieved results should maximize the information covered regarding all aspects of the query.

Ambiguity in the information need can fall into different levels of granularity. For example, the query "Support Vector Machine" might be issued from users looking for tutorials, downloadable implementations, specific formulations (e.g., SVMs for ranking), or theoretical results. The ambiguity for this query is clearly more fine-grained than for the aforementioned query "Jaguar". Learning queries require very fine-grained measures of information diversity. Existing evaluation measures are typically calculated over manually defined information needs (specific to each query) [53, 77, 43, 186], or must rely on live user studies [156]. The query-

specific information needs (also known as information nuggets or subtopics) are either pre-defined and corpus independent [53], or are determined after viewing a set of candidate documents for that query [77]. The granularity of these evaluation measures is then determined by the granularity of manual labeling, and thus can vary from query to query.

For relatively broad topics, existing ML approaches can effectively categorize documents using large topic hierarchies [27, 59, 172, 69, 118]. Recent IR studies have also demonstrated the benefit of using global topic hierarchies to augment retrieval functions [22, 9, 114, 56]. For queries with coarse-grained ambiguities (such as for the query "Jaguar"), it may be sufficient to simply use these hierarchical classification techniques to automatically determine the different information needs of such queries. But it is much more difficult for such techniques to improve search results for queries such as "Support Vector Machine" and for learning queries in general.

More generally, one should optimize metrics that best reflect user utility. Performance measures such as mean average precision and NDCG can be interpreted as ways to model (i.e., approximate) user utility – albeit ones that make very strong independence assumptions regarding the relevance each document.[2] Chapter 4 describes a general learning approach for optimizing utility functions that are sensitive to inter-document overlap and redundancy.

[2]The measures described in Section 2.3 all decompose into a (weighted) sum individual relevances of documents at each rank position.

2.5 Learning from User Interactions

The earliest studies on evaluating and optimizing information retrieval systems commonly used the Cranfield methodology (such as many tasks in TREC [168]), which relies on explicit relevance judgments collected from human experts. Given such a collection of labeled data, the aforementioned supervised learning approaches can then be applied, evaluated, and compared.

Unfortunately, acquiring explicit relevance judgments is quite costly and time consuming, making it difficult to apply at scale for large search services such as commercial search engines. It is also infeasible to collect explicit relevance judgments across a variety of search domains such as patent or medical search. Most importantly, it ignores many other aspects of the usage context. Indeed, some metrics based on human judgments have been shown to not necessarily correlate with more user-centric performance measures [166]. Thus, it can be difficult to use labeled data to generate representative models of user utility. Consequently, collecting usage logs such as clickthrough data has become increasingly popular in recent years.

Implicit feedback offers many benefits. First, it is harvested from usage data such as clickthrough logs, and is thus cheap to acquire and plentiful. It is also naturally representative of the target user population. As such, successfully integrating implicit feedback into the development of retrieval models can greatly improve search performance. For example, one can incorporate implicit user feedback as features into a standard batch learning algorithm [3, 39].

Implicit feedback can be also used instead of explicitly labeled data when modeling and optimizing user utility (cf. [20, 44, 94, 157]). The challenge lies in cor-

rectly interpreting clickthrough results into a quantifiable value for optimization purposes. Despite implicit feedback being noisier than explicitly labeled training data, the sheer quantity of data available will hopefully allow us to learn more effective retrieval models.

Accurate interpretation of usage logs is an area of intense study (see [93] for an overview). The most prevalent issue is that of position bias – that users tend to click more on higher ranked results [83, 71, 88] – which must be addressed when deriving reliable implicit feedback from clickthrough data. Since users typically scan results in rank order, clicking on higher ranked results does not necessarily indicate relevance. For example, one way to leverage usage data as an evaluation metric is by adjusting for bias post-collection [169].

2.5.1 Eliciting Unbiased Feedback

One effective line of approach is to infer pairwise or relative preferences from usage logs. For example, one can interpret a clicked document to be more relevant than an unclicked document presented higher in the ranking [83]. This type of feedback integrates well with the learning methods discussed in Section 2.2.3 that optimize over pairwise preferences, and has been shown to agree with human judged relative preferences [88, 135]. Since users often reformulate queries after finding no satisfactory results from the original query [155, 112], relative preferences can be extended across multiple query formulations: a clicked on document in this "query chain" can be interpreted as being more relevant than unclicked documents presented in servicing earlier query formulations [136].

A more proactive approach to eliciting unbiased feedback is to apply on-line

experiment design in order to preemptively control for position bias. For example, if two competing results were randomly shown in the the original and reversed orders equally often, then clicks might correspond to relative preferences between the two results (i.e., the superior result is clicked on more often) [83, 137, 138, 50].

Of particular relevance to this dissertation are evaluation methods which elicit pairwise preferences over the entire set or ranking of retrieved results. For instance, to elicit whether a user prefers ranking r_1 over r_2, Radlinski et al [140] showed how to present an interleaved ranking of r_1 and r_2 so that clicks indicate which of the two has higher utility (this is described in greater detail in Chapter 7). One advantage of this approach is that it allows for inferring which ranking has higher utility (to the users) without explicitly defining a possibly inaccurate model of user utility.

2.5.2 Towards Interactive Learning

From a machine learning perspective, a major limitation of the aforementioned approaches is that implicit feedback is collected *passively*, e.g. users' clicks are logged on results retrieved by the incumbent retrieval function. Thus, users will never see (and thus never click on and provide feedback for) results not ranked highly by the incumbent retrieval function. A more satisfying approach is to allow the search engine to *actively* choose which results to present in order to optimize for some compromise between providing good service and exploring for better retrieval strategies. For example, given a family of retrieval functions, one can design adaptive approaches that interact with users by choosing which two retrieval functions to compare (e.g., via interleaving as described above), and then use users' clicks to infer unbiased feedback. This motivates the interactive learning

framework, called the *Dueling Bandits Problem*, that is described in Chapter 5.

2.6 Other Related Work

The intersection of machine learning and information retrieval is quite broad and diverse, much of which is beyond the scope of this dissertation. The following provides a brief overview of a few prominent related areas.

2.6.1 Language Modeling

Probabilistic language modeling [188, 107] is a research area with strong ties to both Information Retrieval and Natural Language Processing. The standard premise is relatively simple to state. Given a probabilistic language model $P(d, q)$ of how to generate (i.e., sample) documents and queries, one can then compute the following relevance measure by invoking Bayes rule:

$$P(d|q) = \frac{P(q, d)}{P(q)}, \tag{2.13}$$

One simple intuition to explain (2.13) is the following: the user issuing the query has a particular document in mind that satisfies the information need – but which one is it? The conditional distribution $P(d|q)$ models the conditional probability that any particular document d is the one that satisfies a user that issues query q. An application of Bayes rule shows this to be equivalent to the RHS of (2.13). Note that $P(q)$ is constant, since we're dealing here with a single query. This yields

$$P(d|q) \stackrel{rank}{=} P(d, q), \tag{2.14}$$

which is often computed as $P(q|d)P(d)$ due to computational convenience. Since $P(d)$ is also often assumed to be uniform for all d,[3] this yields

$$P(d|q) \stackrel{rank}{=} P(q|d). \tag{2.15}$$

Note that computing $P(q|d)$ might not require anything more complex than simple vector space models, and in practice often reduces to some kind of frequency counting of the query terms within the document d. One advantage of language modeling approaches is that they offer a principled framework for designing such probabilistic models (which can incorporate smoothing via priors). One disadvantage is that model estimation is typically done via maximum likelihood (as is appropriate for probabilistic generative models), which may not lead to the best retrieval performance as determined by rank-based measures such as NDCG.

Beyond directly applying to the standard retrieval setting, language models are also used to capture salient properties of a corpus, such as its topics (or aspects or clusters) [54, 81, 19]. These methods are often employed when developing diversified retrieval approaches, since modeling the different topics or clusters within a set of candidate documents is often used as a pre-processing step to result diversification.

2.6.2 Relevance Feedback

In the relevance feedback setting, feedback from users or expert judges are used to augment the query in order to generate more informative models of the information

[3]The model $P(d, q)$ is often estimated using a corpus of documents, where each document is assumed to be sampled according to some unknown distribution. Making the further assumption that this sampling occurs i.i.d. immediately suggests that $P(d)$ should be uniform.

need [149]. At a high level, there are three types of relevance feedback: explicit feedback from users or expert judges, implicit feedback collected from users as they interact with the system, and pseudo or blind feedback which is collected without any human response (e.g., by assuming the top results retrieved by an existing search engine tend to be relevant). This feedback is typically collected for individual documents (e.g., feedback that certain documents are relevant to the query), but other forms of feedback are also possible.

Such feedback is then used to build a more refined model of user intent. Well-known methods include vector space model approaches [146, 151], language model approaches [130, 187], as well as query expansion approaches [23, 28, 51, 9]. These motheods all follow the same general motivation: the resulting query intent should be close with respect to a chosen similarity measure to the documents provided by the relevance feedback. One can alternatively employ various approaches such as network analysis techniques[104, 105] to directly compute the quality or relevance of a candidate set of documents without explicitly producing a query/document distance measure.

Interactive learning can be thought of as a way to model how a live retrieval system should gather feedback in an on-line setting. Most prior research on relevance feedback have instead focused either on how to best integrate feedback into an existing model class, or how to best gather feedback (i.e. explore) in order to learn as much as possible. In contrast, interactive learning is concerned with optimizing the entire user experience over time. One particularly important component is how to balance the exploration versus exploitation trade-off – this is discussed further in Chapters 5 and 6.

Part II

Structured Prediction

CHAPTER 3

STRUCTURED PREDICTION AND OPTIMIZING RANK-BASED PERFORMANCE MEASURES

Structured prediction refers to any type of prediction task that is performed jointly over multiple input instances or a single complex input (e.g., a ranking over a set of documents). Rankings are the most common types of structured outputs within information retrieval.

At an abstract level, a structured prediction task is much like a multi-class classification task. Let $\mathcal{X}$ be the space of structured input instances and $\mathcal{Y}$ be the space of structured ouput labels. Each possible structure $\mathbf{y} \in \mathcal{Y}$ (e.g., a parse tree or a ranking) corresponds to one "class", and classifying a new example $\mathbf{x}$ amounts to predicting its correct "class". We can write the structured hypothesis function (which is used to make predictions) as

$$h(\mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} F(\mathbf{x}, \mathbf{y}), \tag{3.1}$$

where $F : \mathcal{X} \times \mathcal{Y} \rightarrow R$, also known as the joint discriminant, measures the quality of predicting $\mathbf{y} \in \mathcal{Y}$ for a given input $\mathbf{x} \in \mathcal{X}$. The hypothesis function h then predicts by choosing the best possible $\mathbf{y}$. For simplicity, we restrict ourselves to discriminants which are parameterized linearly in some large feature space,

$$F(\mathbf{x}, \mathbf{y}|\mathbf{w}) = \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}), \tag{3.2}$$

where $\mathbf{w}$ is the model weight vector, and $\Psi : \mathcal{X} \times \mathcal{Y} \rightarrow R^M$, also known as the joint feature map, is a high dimensional feature map characterizing the compatibility of $\mathbf{x}$ and $\mathbf{y}$, and also captures the structure of the prediction task.

The strength of the joint feature formulation Ψ is that it allows for using features that examine the combined properties of $\mathcal{X}$ and $\mathcal{Y}$ (rather than just $\mathcal{X}$ as

in most conventional machine learning approaches). This way, the number features in Ψ need not depend on $\mathcal{Y}$ at all, leading to a compact feature representation. For example, in part of speech tagging, we can define Ψ as

$$\Psi(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sum_i \phi_1(y^{(i)}, y^{(i-1)}) \\ \\ \sum_i \phi_2(x^{(i)}, y^{(i)}) \end{bmatrix},$$

where ϕ_1 is feature vector describing adjacent part-of-speech tags $y^{(i)}$ and $y^{(i-1)}$ (e.g., one feature for each possible transition, noun/verb, noun/conjunction, etc.), and ϕ_2 is a feature vector describing the assignment of tag $y^{(i)}$ to word $x^{(i)}$ in the input sentence $\mathbf{x}$. This is structurally equivalent to a hidden Markov model, thus allowing $h(\mathbf{x})$ to be computed efficiently via the standard Viterbi algorithm [45, 7, 165, 106].

In recent years, numerous structured prediction approaches have been proposed that span an impressive range of applications including part-of-speech tagging, parsing and segmentation [45, 161, 106, 7, 165, 8], object recognition and stereo vision problems [158, 64, 133, 160, 148, 78], sequence alignment problems in computational biology [177, 142], and clustering [62, 63]. These approaches all exploit known structure in the prediction task, which typically makes prediction (i.e., computing $h(x)$) tractable.

Given an appropriate formulation for Ψ and a learned model $\mathbf{w}$, (3.1) can often be solved using existing algorithms such as the Viterbi algorithm for hidden Markov models, the CYK algorithm for parse trees [165], graph cuts [101, 97] and belief propagation [176, 110] for Markov random fields, and sorting for rankings. Given a training set of labeled inputs $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, training a good model vector w can, in principle, be accomplished using methods such as perceptrons [45], conditional random fields [106], SVMs [160, 165, 159], neural networks [25], and general gra-

dient descent techniques [46, 162]. We build upon the structural SVM framework [164, 165, 87, 89] for developing our approach.

3.1 Optimizing Rank-Based Performance Measures

Commonly used information retrieval performance measures such as MAP and NDCG (see Section 2.3) provide precise optimization goals during training. Unfortunately, small changes in the document output scores do not necessarily change the ranking, thus causing no change in the rank-based performance measures. When rankings do change (due to two document scores swapping in relative magnitude), they cause instantaneous changes in the performance measures. Thus, these rank-based measures are either flat or discontinuous everywhere with respect to a similarity function's model parameters. This causes great difficulty when attempting to optimize via gradient descent approaches [181]. Initial approaches to optimizing these performance measures either used ad-hoc heuristics [29], or optimized over very restricted parameter spaces [123].

In lieu of directly optimizing rank-based measures, a surrogate objective function is often used. Approaches based on boosting optimize an exponential loss upper bound [190, 173], whereas approaches based on SVMs optimize a hinge loss upper bound of performance loss [37, 182, 174]. Both upper bounds are smooth and can be optimized as is or with regularization (such as L2 regularization commonly used in SVMs). We will present an SVM approach for optimizing mean average precision (more precisely, the approach will optimize a convex loss function that is a rigorous upper bound of average precision loss).

Following the supervised learning setup described in Chapter 2, our goal is to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ between an input space $\mathcal{X}$ (e.g., all possible queries)

and output space $\mathcal{Y}$ (e.g., rankings over a corpus). In order to quantify the quality of a prediction, $\hat{\mathbf{y}} = h(\mathbf{x})$, we will use a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re$ that allows us incorporate specific performance measures such as MAP. Given a finite set of training pairs, $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, N\}$, the performance of h on S can be measured by the empirical risk,

$$R_S^{\Delta}(h) = \frac{1}{N} \sum_{i=1}^{N} \Delta(\mathbf{y}_i, h(\mathbf{x}_i)).$$

Since we are focusing on functions which are parametrized by a weight vector $\mathbf{w}$, we can restate the goal as finding the $\mathbf{w}$ which minimizes the empirical risk,

$$R_S^{\Delta}(\mathbf{w}) \equiv R_S^{\Delta}(h(\cdot|\mathbf{w})). \tag{3.3}$$

In the case of learning a ranked retrieval function, $\mathcal{X}$ corresponds to a space of queries, and $\mathcal{Y}$ to a space of (possibly weak) rankings over some corpus of documents $\mathcal{C} = \{d_1, \ldots, d_{|\mathcal{C}|}\}$.

We can define average precision loss as

$$\Delta_{map}(y, \hat{y}) = 1 - \text{MAP}(rank(y), rank(\hat{y})),$$

where $rank(y)$ is a vector of the rank values of each document in $\mathcal{C}$. For example, for a corpus of two documents, $\{d_1, d_2\}$, with d_1 being more relevant than d_2 according to y, we can write $rank(y) = (1, 0)$. We assume that ground truth rankings have two rank values, where relevant documents have rank value 1 and non-relevant documents rank value 0. We further assume that all predicted rankings are complete rankings (no ties).

Let $p = rank(y)$ and $\hat{p} = rank(\hat{y})$. The average precision score is defined as

$$\text{MAP}(p, \hat{p}) = \frac{1}{rel} \sum_{j:p_j=1} Prec@j,$$

where $rel = |\{i : p_i = 1\}|$ is the number of relevant documents, and $Prec@j$ is the percentage of relevant documents in the top j documents in predicted ranking $\hat{y}$. MAP is the mean of the average precision scores of a group of queries.

It remains to develop an appropriate feature formulation of the hypothesis function h (i.e., Ψ) as well as an effective supervised training approach for optimizing MAP. As we shall see in the following, the structure of the joint feature map Ψ is relatively simple. But unlike many other structured prediction learning problems, the challenge here lies in solving the induced optimization problem during training, since MAP is a complex multivariate loss function that is defined over entire rankings of labeled documents.[1]

Comparing MAP with ROCArea and Accuracy

Most learning algorithms optimize for accuracy or ROCArea. While optimizing for these measures might achieve good MAP performance, we use two simple examples to show it can also be suboptimal in terms of MAP.

ROCArea assigns equal penalty to each misordering of a relevant/non-relevant pair. In contrast, MAP assigns greater penalties to misorderings higher up in the predicted ranking. Using our notation, ROCArea can be defined as

$$\mathrm{ROC}(p, \hat{p}) = \frac{1}{rel \cdot (|\mathcal{C}| - rel)} \sum_{i:p_i=1} \sum_{j:p_j=0} \mathbf{1}_{[\hat{p}_i > \hat{p}_j]},$$

where p is the ground truth (weak) ranking, $\hat{p}$ is the predicted ranking, and $\mathbf{1}_{[b]}$ is the indicator function conditioned on predicate b.

[1] In most other structured prediction learning problems, the loss function is fairly simple and can be evaluated independently on each individual prediction (e.g., Hamming loss in sequence labeling problems).

Table 3.1: Comparing MAP and ROCArea: Toy Example and Models

Doc ID	1	2	3	4	5	6	7	8
p	1	0	0	0	0	1	1	0
$rank(h_1(\mathbf{x}))$	8	7	6	5	4	3	2	1
$rank(h_2(\mathbf{x}))$	1	2	3	4	5	6	7	8

Table 3.2: Comparing MAP and ROCArea: Performance of Toy Models

Hypothesis	MAP	ROCArea
$h_1(\mathbf{x})$	0.59	0.47
$h_2(\mathbf{x})$	0.51	0.53

Suppose we have a hypothesis space with only two hypothesis functions, h_1 and h_2, as shown in Table 3.1. These two hypotheses predict a ranking for query x over a corpus of eight documents. Note that $rank(\cdot)$ gives the rank label (so a larger value means more relevant). Table 3.2 shows the MAP and ROCArea scores of h_1 and h_2. Here, a learning method which optimizes for ROCArea would choose h_2 since that results in a higher ROCArea score, but this yields a suboptimal MAP score.

Using a very similar example, we can also demonstrate how optimizing for accuracy might result in suboptimal MAP. Models which optimize for accuracy are not directly concerned with the ranking. Instead, they learn a threshold such that documents scoring higher than the threshold can be classified as relevant and documents scoring lower as non-relevant.

Consider again a hypothesis space with two hypotheses. Table 3.3 shows the predictions of the two hypotheses on a single query $\mathbf{x}$. Table 3.4 shows the MAP and best accuracy scores of $h_1(q)$ and $h_2(q)$. The best accuracy refers to the highest achievable accuracy on that ranking when considering all possible thresholds. For instance, with $h_1(q)$, a threshold between documents 1 and 2 gives 4 errors (documents 6-9 incorrectly classified as non-relevant), yielding an accuracy of 0.64.

Table 3.3: Comparing MAP and Accuracy: Toy Example and Models

Doc ID	1	2	3	4	5	6	7	8	9	10	11
p	1	0	0	0	0	1	1	1	1	0	0
$rank(h_1(x))$	11	10	9	8	7	6	5	4	3	2	1
$rank(h_2(x))$	1	2	3	4	5	6	7	8	9	10	11

Table 3.4: Comparing MAP and Accuracy: Performance of Toy Models

Hypothesis	MAP	Best Acc.
$h_1(q)$	0.56	0.64
$h_2(q)$	0.51	0.73

Similarly, with $h_2(q)$, a threshold between documents 5 and 6 gives 3 errors (documents 10-11 incorrectly classified as relevant, and document 1 as non-relevant), yielding an accuracy of 0.73. A learning method which optimizes for accuracy would choose h_2 since that results in a higher accuracy score, but this yields a suboptimal MAP score.

3.1.1 Structured Prediction Model for Optimizing MAP

We develop our structured prediction model by building off the approach proposed in [85] for optimizing ROCArea. Unlike ROCArea, however, MAP does not decompose linearly in the examples and requires a substantially extended algorithm [182], which we describe in the following sections.

Recall that the ground truth ranking is a weak ranking with two rank values (relevant and non-relevant). Let $\mathcal{C}^x$ and $\mathcal{C}^{\bar{x}}$ denote the set of relevant and non-relevant documents of $\mathcal{C}$ for query $\mathbf{x}$, respectively. Following the notation established in (3.1) and (3.2), the combined feature function we use is

$$\Psi(\mathbf{x}, \mathbf{y}) = \frac{1}{|C^x| \cdot |C^{\bar{x}}|} \sum_{i:d_i \in \mathcal{C}^x} \sum_{j:d_j \in C^{\bar{x}}} \left[y_{ij} \left(\phi(\mathbf{x}, d_i) - \phi(\mathbf{x}, d_j) \right) \right], \tag{3.4}$$

where $\phi : \mathcal{X} \times \mathcal{C} \rightarrow \Re^{\mathcal{N}}$ is a feature mapping function from a query/document pair to a point in $\mathcal{N}$ dimensional space.[2] We represent rankings as a matrix of pairwise orderings, $\mathcal{Y} \subset \{-1, 0, +1\}^{|\mathcal{C}| \times |\mathcal{C}|}$. For any $\mathbf{y} \in \mathcal{Y}$, $y_{ij} = +1$ if d_i is ranked ahead of d_j, and $y_{ij} = -1$ if d_j is ranked ahead of d_i, and $y_{ij} = 0$ if d_i and d_j have equal rank. We consider only matrices which correspond to valid rankings (i.e, obeying antisymmetry and transitivity). Intuitively, Ψ is a summation over the vector differences of all relevant/non-relevant document pairings. Since we assume predicted rankings to be complete rankings, y_{ij} is either $+1$ or -1, and never 0.

Given a learned weight vector $\mathbf{w}$, predicting a ranking (i.e., solving (3.1)) given query $\mathbf{x}$ reduces to picking each y_{ij} to maximize $\mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y})$. As is also discussed in [85], this is attained by sorting the documents by $\mathbf{w}^T \phi(\mathbf{x}, d)$ in descending order.[3] Thus the formulation in (3.1) provides an explicit structure that is compatible with many exist classes of retrieval functions.[4] We will discuss later the choices of ϕ we used for empirical evaluations.

3.2 Training with Structural SVMs

The formulation in (3.4) is very similar to learning a straightforward linear model while training on the pairwise difference of relevant/non-relevant document pairings. Many SVM-based approaches optimize over these pairwise differences (e.g., [32, 75, 85, 29]), although these methods do not optimize for MAP during training.

[2]For example, one dimension might be the number of times the query words appear in the document.

[3]Note that the feature formulation requires knowing the relevance labels, which are unavailable at test time. However, since the argmax is solved by sorting on $\mathbf{w}^T \phi(\mathbf{x}, d)$, the explicit joint feature model is not actually required when making predictions.

[4]Most existing approaches use some method to train or tune $\mathbf{w}$, after which rankings are computed by sorting on $\mathbf{w}^T \phi(\mathbf{x}, d)$. However, a full model is usually left undefined.

Previously, it was not clear how to incorporate non-linear multivariate loss functions such as MAP loss directly into global optimization problems such as SVM training. We now present a method based on structural SVMs [165] to address this problem.

We use the structural SVM formulation, presented in Optimization Problem 4, to learn a $\mathbf{w} \in \mathbb{R}^{\mathcal{N}}$.

Optimization Problem 4. (STRUCTURAL SVM)

$$\min_{\mathbf{w},\xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{N}\xi_i \tag{3.5}$$

$$s.t.\ \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i :$$

$$\mathbf{w}^T\Psi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T\Psi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \tag{3.6}$$

The objective function to be minimized (3.5) is a trade-off between model complexity, $\|\mathbf{w}\|^2$, and a hinge loss relaxation of MAP loss, $\sum \xi_i$. As is usual in SVM training, C is a parameter that controls this trade-off and can be tuned to achieve good performance in different tasks.

For each $(\mathbf{x}_i, \mathbf{y}_i)$ in the training set, a set of constraints of the form in equation (3.6) is added to the optimization problem. Note that $\mathbf{w}^T\Psi(\mathbf{x}, \mathbf{y})$ is exactly the discriminant function $F(\mathbf{x}, \mathbf{y}|\mathbf{w})$ (see equation (3.2)). During prediction, the model chooses the ranking which maximizes the discriminant (3.1). If the discriminant value for an incorrect ranking $\mathbf{y}$ is greater than for the true ranking $\mathbf{y}_i$ (i.e., $F(\mathbf{x}_i, \mathbf{y}|\mathbf{w}) > F(\mathbf{x}_i, \mathbf{y}_i|\mathbf{w})$), then the corresponding slack variable, ξ_i, must be at least $\Delta(\mathbf{y}_i, \mathbf{y})$ in order for that constraint to be satisfied. Therefore, the sum of slacks, $\sum \xi_i$, upper bounds the MAP loss. This is stated formally in Proposition 1.

Algorithm 1 Cutting Plane Training for Structural SVMs.

Input: $S = ((x_1, y_1), \ldots, (x_n, y_N))$, C, ϵ
$\mathcal{W} \leftarrow \emptyset$, $\mathbf{w} = \mathbf{0}$, $\xi_i \leftarrow 0$ for all $i = 1, ..., N$
repeat
  for i=1,...,N **do**
    $\hat{y} \leftarrow \text{argmax}_{\hat{y} \in \mathcal{Y}} \{\Delta_i(\hat{y}) + \langle \mathbf{w}, \Psi(x_i, \hat{y}) \rangle\}$
    if $\langle \mathbf{w}, [\Psi(\bar{\mathbf{x}}_i, y_i) - \Psi(\bar{\mathbf{x}}_i, \hat{y})] \rangle < \Delta_i(\hat{y}) - \xi_i - \epsilon$ **then**
      $\mathcal{W} \leftarrow \mathcal{W} \cup \{\langle \mathbf{w}, [\Psi(x_i, y_i) - \Psi(x_i, \hat{y})] \rangle \geq \Delta_i(\hat{y}) - \xi_i\}$
      $(\mathbf{w}, \xi) \leftarrow \underset{\mathbf{w}, \xi \geq \mathbf{0}}{\text{argmin}} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \frac{C}{N} \sum_{i=1}^{N} \xi_i$ s.t. $\mathcal{W}$
    end if
  end for
until $\mathcal{W}$ has not changed during iteration
return($\mathbf{w}$,ξ)

Proposition 1. *Let $\xi^*(\mathbf{w})$ be the optimal solution of the slack variables for OP 4 for a given weight vector $\mathbf{w}$. Then $\frac{1}{N} \sum_{i=1}^{N} \xi_i$ is an upper bound on the empirical risk $R_S^{\Delta}(\mathbf{w})$.*

Proof. The essential observation is that

$$\xi_i^* = \max \left\{ 0, \max_{\mathbf{y}} \{\Delta(\mathbf{y}_i, \mathbf{y}) - \mathbf{w}^T (\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y}))\} \right\},$$

which is guaranteed to upper bound $\Delta(\mathbf{y}_i, \mathbf{y})$ for $\mathbf{y}$ such that □

Proposition 1 shows that OP 4 learns a ranking function that optimizes an upper bound on MAP error on the training set. And since OP 4 is a quadratic (and thus convex) program it should be straightforward to optimize. Unfortunately, the number of constraints (3.6) is typically very large (often exponential in $|\mathcal{Y}|$), thus making it intractable to enumerate all the constraints to input to a standard SVM solver, let alone optimize.

The key idea is to iteratively construct a working set of constraints $\mathcal{W}$ that is equivalent to the full set of constraints in OP 4 up to a specified precision

ϵ. Starting with an empty $\mathcal{W}$ and $\mathbf{w} = 0$, Algorithm 1 iterates through the training examples. For each example, the argmax in Line 5 finds the most violated constraint of the quadratic program in OP 4. If this constraint is violated by more than ϵ (Line 6), it is added to the working set $\mathcal{W}$ (Line 7) and a new $\mathbf{w}$ is computed by solving the quadratic program over the new $\mathcal{W}$ (Line 8). The algorithm terminates and returns the current $\mathbf{w}$ if $\mathcal{W}$ did not change between iterations.

It is obvious that, for any desired ϵ, the algorithm only terminates when it has found an ϵ-accurate solution since it verifies in Line 8 that none of the constraints of the quadratic program in OP 4 are violated by more than ϵ. It can be shown [165] that Algorithm 1 always terminates in a polynomial number of iterations that is independent of the cardinality of the output space $\mathcal{Y}$.

Theorem 1. *[165] Let* $\bar{R} = \max_i \max_{\mathbf{y}} \|\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y})\|$, $\bar{\Delta} = \max_i \max_{\mathbf{y}} \Delta(\mathbf{y}_i, \mathbf{y})$, *and for any* $\epsilon > 0$*, Algorithm 1 terminates after adding at most*

$$\max\left\{\frac{2n\bar{\Delta}}{\epsilon}, \frac{8C\bar{\Delta}\bar{R}^2}{\epsilon^2}\right\}$$

constraints to the working set $\mathcal{W}$.

In fact, a refined version of Algorithm 1 [86, 87] always terminates after adding at most $O(C\epsilon^{-1})$ constraints to $\mathcal{W}$ (typically $|\mathcal{W}| << 1000$). Note that the number of constraints is not only independent of $|\mathcal{Y}|$, but also independent of the number of training examples N.

However, while the number of iterations is small, the argmax in Line 8 might be expensive to compute. Though closely related to solving the argmax for computing predicitons $h(x)$, it has the additional complication in that we must contend with the additional $\Delta(\mathbf{y}_i, \mathbf{y})$ term. Without the ability to efficiently find the most

violated constraint (i.e., solve $\text{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y}, \mathbf{w})$), the constraint generation procedure is not tractable. We will present an efficient algorithm for finding the most violated constraint in the following section.

3.2.1 Finding the Most Violated Constraint

Using OP 4 and optimizing to ROCArea loss (Δ_{roc}) is addressed in [85]. Computing the most violated constraint for Δ_{map} is more difficult. This is primarily because ROCArea decomposes nicely into a sum of scores computed independently on each relative ordering of a relevant/non-relevant document pair. MAP, on the other hand, does not decompose in the same way as ROCArea.

We first define the following objective function,

$$H(\mathbf{y}|\mathbf{w}) \equiv \Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T\Psi(\mathbf{x}_i, \mathbf{y}) - \mathbf{w}^T\Psi(\mathbf{x}_i, \mathbf{y}_i). \tag{3.7}$$

Note that finding the most violated constraint is equivalent to solving $\text{argmax}_{\mathbf{y}\in\mathcal{Y}} H(y)$.

One useful property of Δ_{map} is its invariance to the swapping of any two documents with equal relevance. For example, if documents d_a and d_b are both relevant, then swapping the positions of d_a and d_b in any ranking does not affect Δ_{map}. By extension, Δ_{map} is invariant to any arbitrary permutation of the relevant documents amongst themselves and of the non-relevant documents amongst themselves. However, this reshuffling will affect the discriminant score, $\mathbf{w}^T\Psi(\mathbf{x}, \mathbf{y})$. This leads us to Observation 1.

Observation 1. *Consider rankings which are constrained by fixing the relevance at each position in the ranking (e.g., the 3rd document in the ranking must be relevant). Every ranking which satisfies the same set of constraints will have the same*

Δ_{map}. *If the relevant documents are sorted by* $\mathbf{w}^T\phi(\mathbf{x}, d)$ *in descending order, and the non-relevant documents are likewise sorted by* $\mathbf{w}^T\phi(\mathbf{x}, d)$*, then the interleaving of the two sorted lists which satisfies the constraints will maximize* H *for that constrained set of rankings.*

Observation 1 implies that in the ranking which maximizes H, the relevant documents will be sorted by $\mathbf{w}^T\phi(\mathbf{x}, d)$, and the non-relevant documents will also be sorted likewise. By first sorting the relevant and non-relevant documents, the problem is simplified to finding the optimal interleaving of two sorted lists. We henceforth assume that the relevant documents and non-relevant documents are both sorted by descending $\mathbf{w}^T\phi(\mathbf{x}, d)$. We will also refer to relevant documents as $\{d_1^x, \ldots d_{|\mathcal{C}^x|}^x\} = \mathcal{C}^x$, and non-relevant documents as $\{d_1^{\bar{x}}, \ldots d_{|\mathcal{C}^{\bar{x}}|}^{\bar{x}}\} = \mathcal{C}^{\bar{x}}$.

We define $\delta_j(i_1, i_2)$, with $i_1 < i_2$, as the change in H from when the highest ranked relevant document *ranked after* $d_j^{\bar{x}}$ is $d_{i_2}^x$ to when it is $d_{i_1}^x$. For $i_2 = i_1 + 1$, we have

$$\delta_j(i, i+1) = \frac{1}{|\mathcal{C}^x|}\left(\frac{j}{j+i} - \frac{j-1}{j+i-1}\right) - \frac{2 \cdot (s_i^x - s_j^{\bar{x}})}{|\mathcal{C}^x| \cdot |\mathcal{C}^{\bar{x}}|}, \tag{3.8}$$

where $s_i = \mathbf{w}^T\phi(\mathbf{x}, d_i)$. The first term in (3.8) is the change in Δ_{map} when the ith relevant document has j non-relevant documents ranked before it, as opposed to $j - 1$. The second term is the change in the discriminant score, $\mathbf{w}^T\Psi(\mathbf{x}, \mathbf{y})$, when y_{ij} changes from $+1$ to -1.

$$\begin{array}{c} \hline \ldots, d_i^x, d_j^{\bar{x}}, d_{i+1}^x, \ldots \\ \ldots, d_j^{\bar{x}}, d_i^x, d_{i+1}^x, \ldots \\ \hline \end{array}$$

Figure 3.1: Optimizing Average Precision: Example for $\delta_j(i, i+1)$

Figure 3.1 gives a conceptual example for $\delta_j(i, i+1)$. The bottom ranking differs from the top only where $d_j^{\bar{x}}$ slides up one rank. The difference in the value

Algorithm 2 Finding the Most Violated Constraint for Optimizing MAP

Input: $\mathbf{w}$, $\mathcal{C}^x$, $\mathcal{C}^{\bar{x}}$
sort $\mathcal{C}^x$ and $\mathcal{C}^{\bar{x}}$ in descending order of $\mathbf{w}^T\phi(x,d)$
$s_i^x \leftarrow \mathbf{w}^T\phi(x,d_i^x)$, $i=1,\ldots,|\mathcal{C}^x|$
$s_i^{\bar{x}} \leftarrow \mathbf{w}^T\phi(x,d_i^{\bar{x}})$, $i=1,\ldots,|\mathcal{C}^{\bar{x}}|$
for $j=1,\ldots,|\mathcal{C}^{\bar{x}}|$ **do**
    $opt_j \leftarrow \text{argmax}_k\, \delta_j(k,|\mathcal{C}^x|+1)$
end for
encode $\hat{\mathbf{y}}$ according to (3.10)
return $\hat{\mathbf{y}}$

of H for these two rankings is exactly $\delta_j(i,i+1)$.

For any $i_1 < i_2$, we can then define $\delta_j(i_1,i_2)$ as

$$\delta_j(i_1,i_2) = \sum_{k=i_1}^{i_2-1} \delta_j(k,k+1), \tag{3.9}$$

or equivalently,

$$\delta_j(i_1,i_2) = \sum_{k=i_1}^{i_2-1} \left[\frac{1}{|\mathcal{C}^x|} \left(\frac{j}{j+k} - \frac{j-1}{j+k-1} \right) - \frac{2\cdot(s_k^x - s_j^{\bar{x}})}{|\mathcal{C}^x|\cdot|\mathcal{C}^{\bar{x}}|} \right].$$

Let $o_1,\ldots,o_{|\mathcal{C}^{\bar{x}}|}$ encode the positions of the non-relevant documents, where $d_{o_j}^x$ is the highest ranked relevant document ranked *after* the jth non-relevant document. Due to Observation 1, this encoding uniquely identifies a complete ranking. We can recover the ranking as

$$y_{ij} = \begin{cases} 0 & \text{if } i=j \\ sign(s_i - s_j) & \text{if } d_i, d_j \text{ equal relevance} \\ sign(o_{j'} - i' - 0.5) & \text{if } d_i = d_{i'}^x, d_j = d_{j'}^{\bar{x}} \\ sign(j' - o_{i'} + 0.5) & \text{if } d_i = d_{i'}^{\bar{x}}, d_j = d_{j'}^x \end{cases}. \tag{3.10}$$

We can now reformulate H into a new objective function,

$$H'(o_1,\ldots,o_{|\mathcal{C}^{\bar{x}}|}|\mathbf{w}) = H(\bar{\mathbf{y}}|\mathbf{w}) + \sum_{k=1}^{|\mathcal{C}^{\bar{x}}|} \delta_k(o_k,|\mathcal{C}^x|+1),$$

where $\bar{y}$ is the true (weak) ranking. Conceptually H' starts with a perfect ranking $\bar{\mathbf{y}}$, and adds the change in H when each successive non-relevant document slides up the ranking.

We can then reformulate the argmax H problem as

$$\operatorname{argmax} H' = \underset{o_1,\ldots,o_{|\mathcal{C}^{\bar{x}}|}}{\operatorname{argmax}} \sum_{k=1}^{|\mathcal{C}^{\bar{x}}|} \delta_k(o_k, |\mathcal{C}^x| + 1) \tag{3.11}$$

such that

$$o_1 \leq \ldots \leq o_{|\mathcal{C}^{\bar{x}}|}. \tag{3.12}$$

Algorithm 2 describes the algorithm used to solve equation (3.11). Conceptually, Algorithm 2 starts with a perfect ranking. Then for each successive non-relevant document, the algorithm modifies the solution by sliding that document up the ranking to locally maximize H' while keeping the positions of the other non-relevant documents constant.

3.2.2 Proof of Correctness

Algorithm 2 is greedy in the sense that it finds the best position of each non-relevant document independently from the other non-relevant documents. In other words, the algorithm maximizes H' for each non-relevant document, $d_j^{\bar{x}}$, without considering the positions of the other non-relevant documents, and thus ignores the constraints of (3.12).

In order for the solution to be feasible, then jth non-relevant document must be ranked after the first $j-1$ non-relevant documents, thus satisfying

$$opt_1 \leq opt_2 \leq \ldots \leq opt_{|\mathcal{C}^{\bar{x}}|}. \tag{3.13}$$

If the solution is feasible, then it clearly solves (3.11). Therefore, it suffices to prove that Algorithm 2 satisfies (3.13). We first prove that $\delta_j(\cdot,\cdot)$ is monotonically decreasing in j.

Lemma 1. *For any $1 \leq i_1 < i_2 \leq |\mathcal{C}^x| + 1$ and $1 \leq j < |\mathcal{C}^{\bar{x}}|$, it must be the case that*

$$\delta_{j+1}(i_1, i_2) \leq \delta_j(i_1, i_2).$$

Proof. Recall from (3.9) that both $\delta_j(i_1, i_2)$ and $\delta_{j+1}(i_1, i_2)$ are summations of $i_2 - i_1$ terms. We will show that each term in the summation of $\delta_{j+1}(i_1, i_2)$ is no greater than the corresponding term in $\delta_j(i_1, i_2)$, or

$$\delta_{j+1}(k, k+1) \leq \delta_j(k, k+1)$$

for $k = i_1, \ldots, i_2 - 1$.

Each term in $\delta_j(k, k+1)$ and $\delta_{j+1}(k, k+1)$ can be further decomposed into two parts (see (3.8)). We will show that each part of $\delta_{j+1}(k, k+1)$ is no greater than the corresponding part in $\delta_j(k, k+1)$. In other words, we will show that both

$$\frac{j+1}{j+k+1} - \frac{j}{j+k} \leq \frac{j}{j+k} - \frac{j-1}{j+k-1} \tag{3.14}$$

and

$$-\frac{2 \cdot (s_k^x - s_{j+1}^{\bar{x}})}{|\mathcal{C}^x| \cdot |\mathcal{C}^{\bar{x}}|} \leq -\frac{2 \cdot (s_k^x - s_j^{\bar{x}})}{|\mathcal{C}^x| \cdot |\mathcal{C}^{\bar{x}}|} \tag{3.15}$$

are true for the aforementioned values of j and k.

It is easy to see that (3.14) is true by observing that for any two positive integers $1 \leq a < b$,

$$\frac{a+1}{b+1} - \frac{a}{b} \leq \frac{a}{b} - \frac{a-1}{b-1},$$

and choosing $a = j$ and $b = j + k$.

The second inequality (3.15) holds because Algorithm 2 first sorts $d^{\bar{x}}$ in descending order of $s^{\bar{x}}$, implying $s^{\bar{x}}_{j+1} \leq s^{\bar{x}}_{j}$.

Thus we see that each term in δ_{j+1} is no greater than the corresponding term in δ_j, which completes the proof. □

The result of Lemma 1 leads directly to the main correctness result of this chapter:

Theorem 2. *In Algorithm 2, the computed values of opt_j satisfy (3.13), implying that the solution returned by Algorithm 2 is feasible and thus optimal.*

Proof. We will prove that

$$opt_j \leq opt_{j+1}$$

holds for any $1 \leq j < |\mathcal{C}^{\bar{x}}|$, thus implying (3.13).

Since Algorithm 2 computes opt_j as

$$opt_j = \operatorname*{argmax}_k \delta_j(k, |\mathcal{C}^x| + 1), \tag{3.16}$$

then by definition of δ_j (3.9), for any $1 \leq i < opt_j$,

$$\delta_j(i, opt_j) = \delta_j(i, |\mathcal{C}^x| + 1) - \delta_j(opt_j, |\mathcal{C}^x| + 1) < 0.$$

Using Lemma 1, we know that

$$\delta_{j+1}(i, opt_j) \leq \delta_j(i, opt_j) < 0,$$

which implies that for any $1 \leq i < opt_j$,

$$\delta_{j+1}(i, |\mathcal{C}^x| + 1) - \delta_{j+1}(opt_j, |\mathcal{C}^x| + 1) < 0.$$

Suppose for contradiction that $opt_{j+1} < opt_j$. Then

$$\delta_{j+1}(opt_{j+1}, |\mathcal{C}^x| + 1) < \delta_{j+1}(opt_j, |\mathcal{C}^x| + 1),$$

which contradicts (3.16). Therefore, it must be the case that $opt_j \leq opt_{j+1}$, which completes the proof. □

3.3 Experiments

The main goal of these experiments is to evaluate whether directly optimizing MAP leads to improved MAP performance compared to conventional SVM methods that optimize a substitute loss such as accuracy or ROCArea. We empirically evaluated using two sets of TREC Web Track queries, one each from TREC 9 and TREC 10 (topics 451-500 and 501-550), both of which used the WT10g corpus. For each query, TREC provides the relevance judgments of the documents.

We generated features using the scores of existing retrieval functions on these queries. While the proposed method is agnostic to the meaning of the features, we chose to use existing retrieval functions as a simple yet effective way of acquiring useful features. As such, these experiments essentially test our method's ability to re-rank the highly ranked documents (e.g., re-combine the scores of these retrieval functions) to improve MAP.

We compare against the best retrieval functions trained on (henceforth *base functions*), as well as against previously proposed SVM methods. Comparing with the best base functions tests our method's ability to learn a useful combination. Comparing with previous SVM methods allows us to test whether optimizing directly for MAP (as opposed to accuracy or ROCArea) achieves a higher MAP score

in practice. The rest of this section describes the base functions and the feature generation method in detail.

3.3.1 Choosing Retrieval Functions

We chose to evaluate using two sets of base functions. For the first set, we generated three indexes over the WT10g corpus using Indri.[5] The first index was generated using the default settings, the second used Porter-stemming, and the last used Porter-stemming and Indri's default stopwords.

For both TREC 9 and TREC 10, we used the description portion of each query and scored the documents using five of Indri's built-in retrieval methods, which are Cosine Similarity, TFIDF, Okapi, Language Model with Dirichlet Prior, and Language Model with Jelinek-Mercer Prior. All parameters were kept as their defaults.

We computed the scores of these five retrieval methods over the three indexes, giving 15 base functions in total. For each query, we considered the scores of documents found in the union of the top 1000 documents of each base function.

For the second set of base functions, we used scores from the TREC 9 [73] and TREC 10 [74] Web Track submissions. We used only the non-manual, non-short submissions from both years. For TREC 9 and TREC 10, there were 53 and 18 such submissions, respectively. A typical submission contained scores of its top 1000 documents.

[5] `http://www.lemurproject.org`

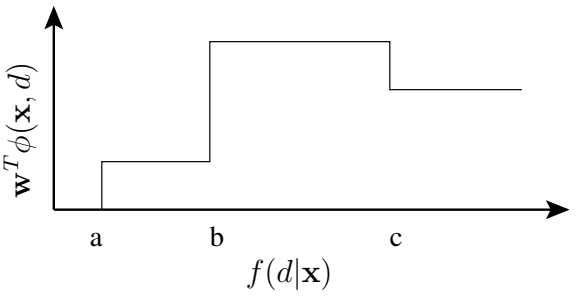

Figure 3.2: Optimizing Average Precision: Example Feature Binning

3.3.2 Generating Features

In order to generate input examples, a concrete instantiation of ϕ must be provided. For each document d scored by a set of retrieval functions $\mathcal{F}$ on query $\mathbf{x}$, we generate the features as a vector

$$\phi(\mathbf{x}, d) = \langle \mathbf{1}_{[f(d|\mathbf{x})>k]} : \forall f \in \mathcal{F}, \forall k \in K_f \rangle,$$

where $f(d|\mathbf{x})$ denotes the score that retrieval function f assigns to document d for query $\mathbf{x}$, and each K_f is a set of real values. From a high level, we are expressing the score of each retrieval function using $|K_f| + 1$ bins.

Since we are using linear kernels, one can think of the learning problem as finding a good piecewise-constant combination of the scores of the retrieval functions. Figure 3.2 shows an example of the feature mapping method. In this example we have a single feature $\mathcal{F} = \{f\}$. Here, $K_f = \{a, b, c\}$, and the weight vector is $\mathbf{w} = \langle w_a, w_b, w_c \rangle$. For any document d and query $\mathbf{x}$, we have

$$\mathbf{w}^T\phi(\mathbf{x}, d) = \begin{cases} 0 & \text{if } f(d|\mathbf{x}) < a \\ w_a & \text{if } a \leq f(d|\mathbf{x}) < b \\ w_a + w_b & \text{if } b \leq f(d|\mathbf{x}) < c \\ w_a + w_b + w_c & \text{if } c \leq f(d|\mathbf{x}) \end{cases}.$$

This is expressed qualitatively in Figure 3.2, where w_a and w_b are positive, and w_c

Table 3.5: Optimizing Average Precision: Dataset Statistics

Dataset	Base Funcs	Features
TREC 9 Indri	15	750
TREC 10 Indri	15	750
TREC 9 Submissions	53	2650
TREC 10 Submissions	18	900

Table 3.6: Comparing $\text{SVM}^{\Delta}_{map}$ with Base Functions

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.242	–	0.236	–
Best Func.	0.204	39/11 **	0.181	37/13 **
2nd Best	0.199	38/12 **	0.174	43/7 **
3rd Best	0.188	34/16 **	0.174	38/12 **

is negative.

We evaluated using four choices of $\mathcal{F}$: the set of aforementioned Indri retrieval functions for TREC 9 and TREC 10, and the Web Track submissions for TREC 9 and TREC 10. For each $\mathcal{F}$ and each function $f \in \mathcal{F}$, we chose 50 values for K_f which are reasonably spaced and which capture the sensitive region of f.

Using the four choices of $\mathcal{F}$, we generated four datasets. Table 3.5 contains statistics of the generated datasets. There are many ways to generate features, and we do not necessarily advocate this particular method over others. This was simply an efficient means to normalize the outputs of different functions and allow for a more expressive model.

Table 3.7: Comparing SVM^{Δ}_{map} with TREC Submissions

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
SVM^{Δ}_{map}	0.290	–	0.287	–
Best Func.	0.280	28/22	0.283	29/21
2nd Best	0.269	30/20	0.251	36/14 **
3rd Best	0.266	30/20	0.233	36/14 **

Table 3.8: Comparing SVM^{Δ}_{map} with TREC Submissions (w/o best)

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
SVM^{Δ}_{map}	0.284	–	0.288	–
Best Func.	0.280	27/23	0.283	31/19
2nd Best	0.269	30/20	0.251	36/14 **
3rd Best	0.266	30/20	0.233	35/15 **

3.3.3 Experiment Results

For each dataset in Table 3.5, we performed 50 trials. For each trial, we train on 10 randomly selected queries, and select another 5 queries at random for a validation set. Models were trained using a wide range of C values. The model which performed best on the validation set was selected and tested on the remaining 35 queries.

The randomization was designed such that all queries were selected to be in the training, validation and test sets the same number of times overall. Using this setup, we performed the same experiments while using our method (SVM^{Δ}_{map}), an SVM optimizing for ROCArea (SVM^{Δ}_{roc}) [85], and a conventional classification SVM (SVM_{acc}) [167]. All SVM methods used a linear kernel. We reported the average performance of all models over the 50 trials.

3.3.4 Comparing SVM^{Δ}_{map} with Base Functions

The first question to answer is, can SVM^{Δ}_{map} learn a model which outperforms the best base functions? Table 3.6 presents the comparison of SVM^{Δ}_{map} with the best Indri base functions. Each column group contains the macro-averaged MAP performance of SVM^{Δ}_{map} or a base function. The W/L columns show the number of queries where SVM^{Δ}_{map} achieved a higher MAP score. Significance tests were performed using the two-tailed Wilcoxon signed rank test. Two stars indicate a significance level of 0.95. All tables displaying the experimental results are structured identically. Here, we find that SVM^{Δ}_{map} significantly outperforms the best base functions.

Table 3.7 shows the comparison when trained on TREC submissions. While achieving a higher MAP score than the best base functions, the performance difference between SVM^{Δ}_{map} the base functions is not significant. Given that many of these submissions use scoring functions which are carefully crafted to achieve high MAP, it is possible that the best performing submissions use techniques which dominate the techniques of the other submissions. As a result, SVM^{Δ}_{map} would not be able to learn a hypothesis which can significantly out-perform the best submission.

Hence, we ran the same experiments using a modified dataset where the features computed using the best submission were removed. Table 3.8 shows the results (note that we are still comparing against the best submission though we are not using it for training). Notice that while the performance of SVM^{Δ}_{map} degraded slightly, the performance was still comparable with that of the best submission.

Table 3.9: Comparing SVM^{Δ}_{map} with SVM^{Δ}_{roc} and SVM_{acc} using Base Functions

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
SVM^{Δ}_{map}	0.242	–	0.236	–
SVM^{Δ}_{roc}	0.237	29/21	0.234	24/26
SVM_{acc}	0.147	47/3 **	0.155	47/3 **
SVM_{acc2}	0.219	39/11 **	0.207	43/7 **
SVM_{acc3}	0.113	49/1 **	0.153	45/5 **
SVM_{acc4}	0.155	48/2 **	0.155	48/2 **

Table 3.10: Comparing $SVM^{\Delta}_{map}$with SVM^{Δ}_{roc} and SVM_{acc} using TREC Submissions

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
SVM^{Δ}_{map}	0.290	–	0.287	–
SVM^{Δ}_{roc}	0.282	29/21	0.278	35/15 **
SVM_{acc}	0.213	49/1 **	0.222	49/1 **
SVM_{acc2}	0.270	34/16 **	0.261	42/8 **
SVM_{acc3}	0.133	50/0 **	0.182	46/4 **
SVM_{acc4}	0.233	47/3 **	0.238	46/4 **

3.3.5 Comparison w/ Conventional SVM Methods

The next question to answer is, does SVM^{Δ}_{map} produce higher MAP scores than conventional SVM methods? Tables 3.9 and 3.10 present the results of SVM^{Δ}_{map}, SVM^{Δ}_{roc}, and SVM_{acc} when trained on the Indri retrieval functions and TREC submissions, respectively. Table 3.11 contains the corresponding results when trained on the TREC submissions without the best submission.

To start with, the results indicate that SVM_{acc} was not competitive with SVM^{Δ}_{map} and SVM^{Δ}_{roc}, and at times underperformed dramatically. As such, we tried several approaches to improve the performance of SVM_{acc}.

3.3.6 Alternate SVM$_{acc}$ Methods

One issue which may cause SVM$_{acc}$ to underperform is the severe imbalance between relevant and non-relevant documents. The vast majority of the documents are not relevant. SVM$_{acc2}$ addresses this problem by assigning more penalty to false negative errors. For each dataset, the ratio of the false negative to false positive penalties is equal to the ratio of the number non-relevant and relevant documents in that dataset. Tables 3.9, 3.10 and 3.11 indicate that SVM$_{acc2}$ still performs significantly worse than SVM$^{\Delta}_{map}$.

Another possible issue is that SVM$_{acc}$ attempts to find just one discriminating threshold b that is query-invariant. It may be that different queries require different values of b. Having the learning method trying to find a good b value (when one does not exist) may be detrimental.

We took two approaches to address this issue. The first method, SVM$_{acc3}$, converts the retrieval function scores into percentiles. For example, for document d, query q and retrieval function f, if the score $f(d|q)$ is in the top 90% of the scores $f(\cdot|q)$ for query q, then the converted score is $f'(d|q) = 0.9$. Each K_f contains 50 evenly spaced values between 0 and 1. Tables 3.9, 3.10 and 3.11 show that the performance of SVM$_{acc3}$ was also not competitive with SVM$^{\Delta}_{map}$.

The second method, SVM$_{acc4}$, normalizes the scores given by f for each query. For example, assume for query q that f outputs scores in the range 0.2 to 0.7. Then for document d, if $f(d|q) = 0.6$, the converted score would be $f'(d|q) = (0.6 - 0.2)/(0.7 - 0.2) = 0.8$. Each K_f contains 50 evenly spaced values between 0 and 1. Again, Tables 3.9, 3.10 and 3.11 show that SVM$_{acc4}$ was not competitive with SVM$^{\Delta}_{map}$

Table 3.11: Comparing SVM^{Δ}_{map} with SVM^{Δ}_{roc} and SVM_{acc} using TREC Submissions (w/o Best)

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
SVM^{Δ}_{map}	0.284	–	0.288	–
SVM^{Δ}_{roc}	0.274	31/19 **	0.272	38/12 **
SVM_{acc}	0.215	49/1 **	0.211	50/0 **
SVM_{acc2}	0.267	35/15 **	0.258	44/6 **
SVM_{acc3}	0.133	50/0 **	0.174	46/4 **
SVM_{acc4}	0.228	46/4 **	0.234	45/5 **

3.3.7 MAP vs ROCArea

SVM^{Δ}_{roc} performed much better than SVM_{acc} in our experiments. When trained on Indri retrieval functions (see Table 3.9), the performance of SVM^{Δ}_{roc} was slight, though not significantly, worse than the performances of SVM^{Δ}_{map}. However, Table 3.10 shows that SVM^{Δ}_{map} did significantly outperform SVM^{Δ}_{roc} when trained on the TREC submissions.

Table 3.11 shows the performance of the models when trained on the TREC submissions with the best submission removed. The performance of most models degraded by a small amount, with SVM^{Δ}_{map} still having the best performance.

3.4 Discussion

The proposed SVM^{Δ}_{map} method provides a principled approach and avoids difficult to control heuristics. This make it conceptually just as easy to optimize SVMs for MAP as was previously possible only for Accuracy and ROCArea. The computational cost for training is very reasonable in practice. Since other methods typically require tuning multiple heuristics, we also expect to train fewer models

before finding one which achieves good performance.

The learning framework is fairly general. A natural extension would be to develop methods to optimize for other important IR measures, such as Normalized Discounted Cumulative Gain [26, 25, 29, 82, 175] and Mean Reciprocal Rank. One such follow-up study [35] showed that optimizing for a compromise between these different measures can yield more robust performance.

3.4.1 Other Feature Structure Formulations

The joint feature formulation used for optimizing average precision (3.4) has two useful properties. First, it provides an explicit model that can quantify the quality of any ranking (which is required for structural SVM training). Second, making predictions (i.e., finding the best ranking) reduces to sorting using the model's scores on individual documents, thus making the approach compatible for optimizing over many existing classes of retrieval functions.

There are other feature structures that also satisfy the two aforementioned properties. For example, let each $\mathbf{y} \in \mathcal{Y}$ be encoded as a permutation (i.e., y_i is the rank position of the ith document according the permutation $\mathbf{y}$). Then one can consider the following joint feature formulation that was proposed independently of our work [37, 35],

$$\mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}) = \sum_j \alpha(y_j) \mathbf{w}^T \phi(\mathbf{x}, d_j), \tag{3.17}$$

where $\phi(\mathbf{x}, d_j)$ again denotes a feature vector describing document $d_j \in \mathbf{x}$, and α is a non-negative, monotonically non-increasing function defined over the positive integers (e.g., $\alpha(i) = 1/\log(i+1)$). It is straightforward to see that making

predictions (i.e., solving $\text{argmax}_{\mathbf{y}} \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y})$) also reduces to sorting by $\mathbf{w}^T \phi(\mathbf{x}, d)$. This joint feature structure was first proposed for optimizing NDCG and mean reciprocal rank [37].

So how do the two formulations (3.4) and (3.17) compare? While the argmax problem for both reduces to a simple sort, the two formulations differ in how they model or quantify the quality of matching a set of input documents with any given ranking. For instance, (3.4) models the quality of a ranking based purely on the *relative* difference of document scores (e.g., $\mathbf{w}^T \phi(\mathbf{x}, d_i) - \mathbf{w}^T \phi(\mathbf{x}, d_j)$) between a relevant and non-relevant document pair. As such, the quality of predicting any given ranking depends only on whether relevant documents are ranked ahead of non-relevant documents, and can only be explicitly computed when given the true relevance labels. For example, swapping the rank positions of two adjacent relevant documents does not change the joint discriminant score.

On the other hand, (3.17) computes quality based purely on the document scores $\mathbf{w}^T \phi(\mathbf{x}, d)$ weighted by the α value assigned to each rank position. For example, if $\alpha(i) = 0$ for all i greater than some cuttoff (e.g., $i > 10$), then swapping the rank positions of any two documents ranked below that cutoff will not change the joint discriminant score. In this setting, finding the most violated constraint reduces to a linear assignment problem [37], which can be substantially more expensive to solve than the greedy approach described in Section 3.2 for (3.4).

It is worth noting that neither feature formulation is more "correct" than the other in the sense that they are both structured prediction models for measuring ranking quality. The more general issue at play here is understanding and resolving potential mismatches between the structure of the joint feature map Ψ and the structure of the loss function Δ.

3.4.2 Alternative Approaches: Smooth Approximations

Instead of optimizing a convex upper bound on performance loss, an alternative line of approach is to optimize for a smoothed version of the rank-based performance measures. Smoothing techniques are generally popular when dealing with discontinuous or high-curvature functions. While typically not convex, one benefit of using smoothed approximations over convex upper bounds is that they might better approximate the actual performance measure.

Consider the simple case of optimizing over a single query with documents $x_1, \ldots, x_n$ (extending to multiple queries is straightforward). For a fixed model class h with parameters $\mathbf{w}$, let $U(\sigma)$ denote a generic rank-based measure (e.g., MAP or NDCG) of a ranking σ of the documents $x_1, \ldots, x_n$. We can state the goal as finding the $\mathbf{w}$ which maximizes $U(\mathbf{w}) \equiv U(sort\{h(x_1|\mathbf{w}), \ldots, h(x_n|\mathbf{w})\})$.[6] Since $U(\mathbf{w})$ is discontinuous, the approaches described in the following define a smoothed objective function $\hat{U}(\mathbf{w})$ such that $\partial\hat{U}/\partial\mathbf{w}$ exists and is efficiently computable, and that maximizing $\hat{U}(\mathbf{w})$ will (approximately) maximize $U(\mathbf{w})$.

Two general classes of approaches are (a) explicitly defining a globally smooth approximation and (b) defining a gradient for some implicit smooth approximation (since the gradient is all that is required for gradient descent optimization techniques).

[6] By defining the loss as $\Delta = 1 - U(\mathbf{w})$, minimizing the empirical risk $R_S^{\Delta}(h)$ (3.3) becomes equivalent to maximizing $U(\mathbf{w})$.

Global Smoothing

Perhaps the most straightforward approach is to use a model which adds uncertainty such that each document's output score is spread across a distribution. This can be used to create "soft" versions of the otherwise discontinuous rank-based performance measures. Then we can define a $\hat{U}$ as the expectation of U over the uncertainty, i.e.,

$$\hat{U} = \int U(sort\{s_1, \ldots, s_n\}) ds_1 ds_2 \ldots ds_n,$$

where s_i is a random variable corresponding to the score of the ith document. When the uncertainty is Gaussian distributed, $\hat{U}$ can be naturally optimized using Gaussian processes [163, 72], which have been shown to perform well in practice. One drawback of this approach is that it is not necessarily compatible with existing classes of retrieval functions, which typically sort using a single score per document. Using a single score essentially amounts to computing the maximum likelihood value, i.e.,

$$U(sort\{s_1, \ldots, s_n\}) \text{ s.t. } (s_1, \ldots s_n) = \operatorname{argmax} P(s_1, \ldots, s_n),$$

as opposed to the expected value.

Of course, as the uncertainty (or variance) of the document scores approach zero, the expectation converges to the maximum likelihood value since the distribution of documentscores becomes more and more concentrated. This implies that $\hat{U} \to U$ as the uncertainty of the document scores approaches zero. On the other hand, the less smooth the objective function, the more susceptible the optimization procedure is to local optima. This motivates an iterative training procedure which solves a sequence of optimization problems that gradually reduce the variance [38].

One can also take a more top-down approach to defining probability distributions over rankings [30, 171] rather than over individual document scores (which is a special type of probability distributions over rankings). Different variations will yield different objective functions. In general, these approaches can be effective so long as the smoothed objective is easily differentiable and well approximates the original discontinuous rank-based measure.

Gradient Smoothing

Defining conceptually satisfying global approximations to rank-based measures can be difficult. However, gradient descent techniques do not require an explicitly defined objective function, but rather just a well-behaved gradient definition. Focusing on gradient definitions reduces the problem to finding nice local approximations.

For instance, the LambdaRank method computes a gradient on the document output scores [25, 24]. Assuming the scoring function is C^1-continuous, then simple chain rule will yield a gradient in the model parameter space. In practice, one might assume that the gradient can be decomposed additively into pairwise gradient, which can be defined as

$$\lambda_{ij} = \Delta U(i,j) \left(\frac{1}{1 + e^{s_i - s_j}} \right), \tag{3.18}$$

where s_i and s_j are the output scores for documents x_i and x_j (where x_i is more relevant than x_j), and $\Delta U(i,j)$ is the change in U when documents i and j swap rank positions. A smoothing function is used which decreases in magnitude as the s_i increases relative to s_j. The total derivative for a single document output score

can then be written as

$$\lambda_i \equiv \sum_{j \in D_i^-} \lambda_{ij} - \sum_{j \in D_i^+} \lambda_{ji}, \tag{3.19}$$

where D_i^+ and D_i^- denote sets containing documents that are more relevant and less relevant than document i. LambdaRank has been empirically shown to find local optima for many standard rank-based measures when using neural net function classes [181, 58].

CHAPTER 4

DIVERSIFIED RETRIEVAL AS STRUCTURED PREDICTION

Diversified retrieval is a growing research area within the Information Retrieval community [31, 186, 40, 156, 189, 43, 92]. However, most learning-to-rank approaches cannot adequately model information diversity since they were developed for optimizing conventional ranking models which evaluate each document independently (e.g., by sorting using the document scores). Indeed, several recent studies on diversified retrieval (cf. [186, 156]) emphasized the need to model inter-document dependencies such as information redundancy, which is fundamentally a structured prediction problem. Those machine learning approaches that do consider diversity either cannot explicitly learn to optimize for the task-specific evaluation criterion [55, 40, 191] or are limited to only a single query and cannot generalize effectively [139]. The following describes a general machine learning approach of how to diversify with respect to a given dataset of queries and documents labeled to reflect the underlying information diversity [184].

4.1 The Learning Problem

In this learning setting, we assume that each query is associated with a set of candidate documents and a set of subtopics (which may be distinct to that query). Let $\mathbf{x} = \{x_1, \ldots, x_n\}$ denote the set of candidate documents for a query, and let $\mathbb{T} = \{T_1, \ldots, T_n\}$ be defined such that topic set T_j contains the subtopics covered by document $x_j \in \mathbf{x}$. Topic sets may overlap. The goal then is to select a subset $\mathbf{y}$ of K documents from $\mathbf{x}$ which maximizes subtopic coverage.

If the topic sets $\mathbb{T}$ were known at test time, then we can formulate making

predictions as solving the following optimization problem,

$$\underset{s \subset \mathbf{y}, |s| \leq K}{\operatorname{argmax}} \left| \bigcup_{j \in s} T_j \right|, \tag{4.1}$$

where K is a pre-specified retrieval set size (e.g., $K = 10$). Naively computing the globally optimal solution takes n choose K time, which we consider intractable for most values of K. However, this problem is an instance of the budgeted max coverage problem (and more generally is a budgeted submodular optimization problem) [95]. For such optimization problems, the straightforward greedy selection process (iteratively choosing the document that myopically maximizes the number of subtopics covered) yields a solution that is guaranteed to achieve at least $(1 - 1/e)OPT$, where OPT is the number of subtopics covered by the optimal solution,[1] and typically performs much better than the worst-case lower bound.

However, the topic sets of a candidate set are not known, nor is the set of all possible topics known. Only a set of training examples of the form $(\mathbf{x}^{(i)}, \mathbb{T}^{(i)})$ is assumed available, and our goal is to find a good function for predicting $\mathbf{y}$ in the absence of $\mathbb{T}$. This in essence is the learning problem.

Following the supervised learning setup described in Chapter 2, we formulate our task as learning a hypothesis function $h : \mathcal{X} \to \mathcal{Y}$ to predict a $\mathbf{y}$ when given $\mathbf{x}$. We quantify the quality of a prediction by considering a loss function $\Delta : \mathcal{T} \times \mathcal{Y} \to \Re$ which measures the penalty of choosing $\mathbf{y}$ when the topics to be covered are those in $\mathbb{T}$. Given a set of training examples, $S = \{(\mathbf{x}^{(i)}, \mathbb{T}^{(i)}) \in \mathcal{X} \times \mathcal{T} : i = 1, \ldots, N\}$, the strategy is to find a function h which minimizes the empirical risk,

$$R_S^{\Delta}(h) = \frac{1}{N} \sum_{i=1}^{N} \Delta(\mathbb{T}^{(i)}, h(\mathbf{x}^{(i)})).$$

In order to encourage diversity, the loss function $\Delta(\mathbb{T}, \mathbf{y})$ is defined to be the

[1] See Appendix A for more details.

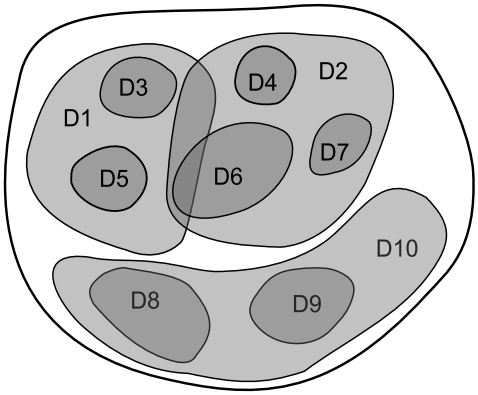

Figure 4.1: Visualization of Documents Covering Subtopics

weighted percentage of distinct subtopics in $\mathbb{T}$ not covered by $\mathbf{y}$, although other formulations are possible.

Following the notation established in (3.1) and (3.2), we can write the hypothesis function as

$$h(\mathbf{x}, \mathbf{w}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}).$$

The feature representation Ψ must enable meaningful discrimination between high quality and low quality predictions. As such, different feature representations may be appropriate for different retrieval settings. We discuss some possible extensions in Section 4.5.

4.2 Maximizing Word Coverage

Figure 4.1 depicts an abstract visualization of the prediction problem. The sets represent candidate documents $\mathbf{x}$ of a query, and the area covered by each set is the "information" (represented as subtopics $\mathbb{T}$) covered by that document. If $\mathbb{T}$ were known, we could use a greedy method to find a solution with high subtopic diversity. For $K = 3$, the optimal solution in Figure 4.1 is $\mathbf{y} = \{D1, D2, D10\}$. In

general however, the subtopics are unknown. We instead assume that the candidate set contains discriminating features which separates subtopics from each other. In this case the features will be derived primarily using word frequencies.

As a proxy for explicitly covering subtopics, we formulate the discriminant Ψ based on weighted word coverage. Intuitively, covering more (distinct) words should result in covering more subtopics. The relative importance of covering any word can be modeled using features describing various aspects of word frequencies within documents in $\mathbf{x}$. We make no claims regarding any generative models relating topics to words, but rather simply assume that word frequency features are highly discriminative of subtopics within $\mathbf{x}$.

4.2.1 Joint Feature Formulation: A Simple Example

Let $V(\mathbf{y})$ denote the union of words contained in the documents of the predicted subset $\mathbf{y}$, and let $\phi(v, \mathbf{x})$ denote the feature vector describing the frequency of word v amongst documents in $\mathbf{x}$. We can then write Ψ as

$$\Psi(\mathbf{x}, \mathbf{y}) = \sum_{v \in V(\mathbf{y})} \phi(v, \mathbf{x}). \tag{4.2}$$

Given a model vector $\mathbf{w}$, the benefit of covering word v in candidate set x is $\mathbf{w}^T \phi(v, \mathbf{x})$. This benefit is realized when a document in $\mathbf{y}$ contains v, i.e., $v \in V(\mathbf{y})$. We use the same model weights for all words. Following the notation established in Chapter 3, a prediction is made by choosing $\mathbf{y}$ to maximize (3.1).

This formulation yields two properties which enable optimizing for diversity. First, covering a word twice provides no additional benefit. Second, the feature vector $\phi(v, \mathbf{x})$ is computed using other documents in the candidate set. Thus,

This word appears ...
... in a document in $\mathbf{y}$.
... at least 5 times in a document in $\mathbf{y}$.
... with frequency at least 5% in a document in $\mathbf{y}$.
... in the title of a document in $\mathbf{y}$.
... within the top 5 TFIDF of a document in $\mathbf{y}$.

Figure 4.2: Maximizing Word Coverage: Examples of Importance Criteria

diversity is measured locally rather than relative to the whole corpus. Both properties are absent from conventional ranking methods which evaluate each document individually.

In practical applications, a more sophisticated Ψ may be more appropriate. We develop the discriminant by addressing two criteria: how well a document covers a word, and how important it is to cover a word in $\mathbf{x}$.

4.2.2 How Well a Document Covers a Word

In the simple example (4.2), a single word set $V(\mathbf{y})$ is used, and all words that appear at least once in $\mathbf{y}$ are included. However, documents do not cover all words equally well, which is something not captured in (4.2). For example, a document which contains 5 instances of the word "lion" might cover the word better than another document which only contains 2 instances.

Instead of using only one $V(\mathbf{y})$, one can use L such word sets $V_1(\mathbf{y}), \ldots, V_L(\mathbf{y})$. Each word set $V_\ell(\mathbf{y})$ contains only words satisfying certain importance criteria. These importance criteria can be based on properties such as appearance in the title, the term frequency in the document, and having a high TFIDF value in the document [150]. Figure 4.2 contains examples of importance criteria. For example, if importance criterion ℓ requires appearing at least 5 times in a document, then

The word v has ...
... a $\lvert D_1(v)\rvert/n$ ratio of at least 40%
... a $\lvert D_2(v)\rvert/n$ ratio of at least 50%
... a $\lvert D_\ell(v)\rvert/n$ ratio of at least 25%

Figure 4.3: Maximizing Word Coverage: Examples of Document Frequency Features

$V_\ell(\mathbf{y})$ will be the set of words which appear at least 5 times in some document in $\mathbf{y}$. The most basic criterion simply requires appearance in a document, and using only this criterion will result in (4.2).

In practice, one can use a separate feature vector $\phi_\ell(v, \mathbf{x})$ for each importance level. We will describe ϕ_ℓ in greater detail section to follow. We can thus define Ψ from (3.1) to be the vector composition of all the ϕ_ℓ vectors,

$$\Psi(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sum_{v \in V_1(\mathbf{y})} \phi_1(v, \mathbf{x}) \\ \vdots \\ \sum_{v \in V_L(\mathbf{y})} \phi_L(v, \mathbf{x}) \\ \sum_{i=1}^{n} y_i \psi(x_i, \mathbf{x}) \end{bmatrix}. \tag{4.3}$$

The last feature vector $\psi(x, \mathbf{x})$ encodes any salient document properties which are not captured at the word level (e.g., "this document received a high score with an existing ranking function").

4.2.3 The Importance of Covering a Word

The feature vectors $\phi_1(v, \mathbf{x}), \ldots, \phi_L(v, \mathbf{x})$ encode the benefit of covering a word, and can be defined in many ways. Here we show a feature definition based primarily on frequency information in $\mathbf{x}$.

Let $D_\ell(v)$ denote the set of documents in $\mathbf{x}$ which cover word v at importance

Algorithm 3 Greedy subset selection by maximizing weighted word coverage

Input: $\mathbf{w}$, $\mathbf{x}$
Initialize solution $\hat{\mathbf{y}} \leftarrow \emptyset$
for $k = 1, \ldots, K$ **do**
    $\hat{x} \leftarrow \text{argmax}_{x:x \notin \hat{\mathbf{y}}} \mathbf{w}^T \Psi(\mathbf{x}, \hat{\mathbf{y}} \cup \{d\})$
    $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} \cup \{\hat{x}\}$
end for
return $\hat{\mathbf{y}}$

level ℓ. For example, if the importance criterion is "appears at least 5 times in the document", then $D_\ell(v)$ is the set of documents that have at least 5 copies of v. This is, in a sense, a complementary definition to $V_\ell(\mathbf{y})$. One can use thresholds on the ratio $|D_\ell(v)|/n$ to define feature values of $\phi_\ell(v, \mathbf{x})$ that describe word v at different importance levels. Figure 4.3 describes examples of such features.

4.2.4 Making Predictions

Putting the formulation together, $\mathbf{w}_\ell^T \phi_\ell(v, \mathbf{x})$ denotes the benefit of covering word v at importance level ℓ, where $\mathbf{w}_\ell$ is the sub-vector of $\mathbf{w}$ which corresponds to ϕ_ℓ in (4.3). A word is only covered at importance level ℓ if it appears in $V_\ell(\mathbf{y})$. The goal then is to select K documents which maximize the aggregate benefit.

Similar to the problem of maximizing subtopic coverage (when the subtopic assignments are known), maximizing the aggregate benefit,

$$h(\mathbf{x}|\mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}}\, \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}),$$

is an instance of the budgeted max coverage problem [95, 79]. Algorithm 3 describes the myopic greedy algorithm which iteratively selects the document with highest marginal gain, which is known to have a $(1 - 1/e)$-approximation bound. Appendix A contains an analysis that proves this performance guarantee.

In summary, we have replaced the problem of maximizing subtopic coverage with the surrogate problem of maximizing weighted word coverage. The surrogate problem does not require human annotations when making predictions, but rather relies on having expressive feature and parameter spaces that can be tuned to the particular retrieval domain. The following section describes a machine learning approach based on structural SVMs for learning the appropriate $\mathbf{w}$ such that predictions made by the model are aligned with the original problem of maximizing subtopic coverage (with respect to the task-specific annotated training data).

4.3 Training with Structural SVMs

For a given training set $S = \{(\mathbb{T}^{(i)}, \mathbf{x}^{(i)})\}_{i=1}^{N}$, we again use the structural SVM formulation to train the model parameters (see Section 3.2 for a more detailed discussion). We restate the optimization problem in slightly modified form using a loss function Δ that assumes the ground truth labelings to have a different format than the model predictions.

Optimization Problem 5. (Structural SVM)

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \tag{4.4}$$

s.t. $\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}^{(i)}$:

$$\mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \geq \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{y}) + \Delta(\mathbb{T}^{(i)}, \mathbf{y}) - \xi_i \tag{4.5}$$

The objective function (4.4) is a tradeoff between model complexity, $\|\mathbf{w}\|^2$, and a hinge loss relaxation of the training loss for each training example, $\sum \xi_i$, and

the tradeoff is controlled by the parameter C. The loss function Δ is typically a coverage loss (e.g., the amount of subtopics not covered). The $\mathbf{y}^{(i)}$ in the constraints (4.5) is the prediction which minimizes $\Delta(\mathbb{T}^{(i)}, \mathbf{y}^{(i)})$, and can be chosen via greedy selection (since choosing optimal prediction can be intractable). The cutting plane algorithm described in Algorithm 1 from Section 3.2 can be used to solve OP 5 efficiently. As in the case of optimizing average precision in Section 3.1, we require a method for finding the most violated constraint (Line 5), or solving for each training example

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \Delta(\mathbb{T}^{(i)}, \mathbf{y}) + \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}), \tag{4.6}$$

Unfortunately, solving (4.6) exactly can be intractable. A straightforward approach is to use the same myopic greedy algorithm that is used for inference. Although the rigorous theoretical results are no longer guaranteed to hold (such as solving original learning problem to within a specified accuracy ϵ), the overall cutting plane training procedure described in Algorithm 1 is still efficient [64]. From an optimization perspective, the cutting plane training procedure is effectively ignoring some of the constraints in the original SVM learning problem. This may cause the model to underfit. We explore this issue empirically in the following section.

4.4 Experiments

We conducted an empirical evaluation using the TREC 6-8 Interactive Track Queries.[2] Relevant documents are labeled using subtopics. For example, query 392 asked human judges to identify different applications of robotics in the world

[2] `http://trec.nist.gov/`

today, and they identified 36 subtopics among the results such as nanorobots and using robots for space missions.

The 17 queries we used are 307, 322, 326, 347, 352, 353, 357, 362, 366, 387, 392, 408, 414, 428, 431, 438, and 446. Three of the original 20 queries were discarded due to having small candidate sets, making them uninteresting for our experiments. Following the setup in [186], candidate sets only include documents which are relevant to at least one subtopic. This decouples the diversity problem, which is the focus of this study, from the relevance problem. In practice, approaches like ours might be used to post-process the results of a commercial search engine. We also performed Porter stemming and stop-word removal.

We used a 12/4/1 split to generate the training, validation and test sets, respectively. We trained using C values varying from 1e-5 to 1e3. The best C value is then chosen on the validation set, and evaluated on the test query. We permuted the train/validation/test splits until all 17 queries were chosen once for the test set. Candidate sets contain on average 45 documents, 20 subtopics, and 300 words per document. We set the retrieval size to $K = 5$ since some candidate sets contained as few as 16 documents.

We compared against Okapi [144], and Essential Pages [156]. Okapi is a conventional retrieval function which evaluates the relevance of each document individually and does not optimize for diversity. Like our method, Essential Pages also optimizes for diversity by selecting documents to maximize weighted word coverage (but based on a fixed, rather than a learned, model). In their model, the benefit of document x_i covering a word v is defined to be

$$TF(v, x_i) \log \left(\frac{1}{DF(v, \mathbf{x})} \right),$$

where $TF(v, x_i)$ is the term frequency of v in x_i and $DF(v, \mathbf{x})$ is the document

Subtopic	# Docs	Weight
t_1	1	1/6
t_2	2	1/3
t_3	2	1/2

Figure 4.4: Weighted Subtopic Loss Example

frequency of v in $\mathbf{x}$.

For these experiments, the loss function was defined to be the weighted percentage of subtopics not covered. For a given candidate set, each subtopic's weight is proportional to the number of documents that cover that subtopic. An example is given in Figure 4.4. This is attractive since it assigns a high penalty to not covering a popular subtopic. It is also compatible with our discriminant since frequencies of important words will vary based on the distribution of subtopics.

The small quantity of TREC queries makes some evaluations difficult, so we also generated a larger synthetic dataset of 100 candidate sets. Each candidate set has 100 documents covering up to 25 subtopics. Each document samples 300 words independently from a multinomial distribution over 5000 words. Each document's word distribution is a mixture of its subtopics' distributions. We used this dataset to evaluate how performance changes with retrieval size K. We used a 15/10/75 split for training, validation, and test sets.

4.4.1 Experiment Results

We evaluated using two versions of the proposed method: $\text{SVM}^{\Delta}_{div}$ which uses term frequencies and title words to define importance criteria (how well a document covers a word), and $\text{SVM}^{\Delta}_{div2}$ which in addition also uses TFIDF. $\text{SVM}^{\Delta}_{div}$ and

Table 4.1: Diversified Retrieval: Performance on TREC Dataset ($K = 5$)

Method	Loss
Random	0.469
Okapi	0.472
Unweighted Model	0.471
Essential Pages	0.434
$\text{SVM}^{\Delta}_{div}$	0.349
$\text{SVM}^{\Delta}_{div2}$	0.382

Table 4.2: Diversified Retrieval: Per Query Comparison on TREC Dataset ($K = 5$)

Method Comparison	Win / Tie / Lose
$\text{SVM}^{\Delta}_{div}$vs Essential Pages	14 / 0 / 3 **
$\text{SVM}^{\Delta}_{div2}$vs Essential Pages	13 / 0 / 4
$\text{SVM}^{\Delta}_{div}$vs $\text{SVM}^{\Delta}_{div2}$	9 / 6 / 2

$\text{SVM}^{\Delta}_{div2}$ use roughly 2000 and 3000 features, respectively.

Table 4.1 shows the performance results on TREC queries. We also included the performance of randomly selecting 5 documents as well as an unweighted word coverage model (all words give equal benefit when covered). Only Essential Pages, $\text{SVM}^{\Delta}_{div}$ and $\text{SVM}^{\Delta}_{div2}$ performed better than random.

Table 4.2 shows the per query comparisons between $\text{SVM}^{\Delta}_{div}$, $\text{SVM}^{\Delta}_{div2}$ and Essential Pages. Two stars indicate 95% significance using the Wilcoxon signed rank test. While the comparison is not completely fair since Essential Pages was designed for a slightly different setting, it demonstrates the benefit of automatically fitting a retrieval function to the specific task at hand.

Despite having a richer feature space, $\text{SVM}^{\Delta}_{div2}$ performs worse than $\text{SVM}^{\Delta}_{div}$. One possibility is that the top TFIDF words do not discriminate between subtopics. These words are usually very descriptive of the query as a whole, and thus may appear in all subtopics.

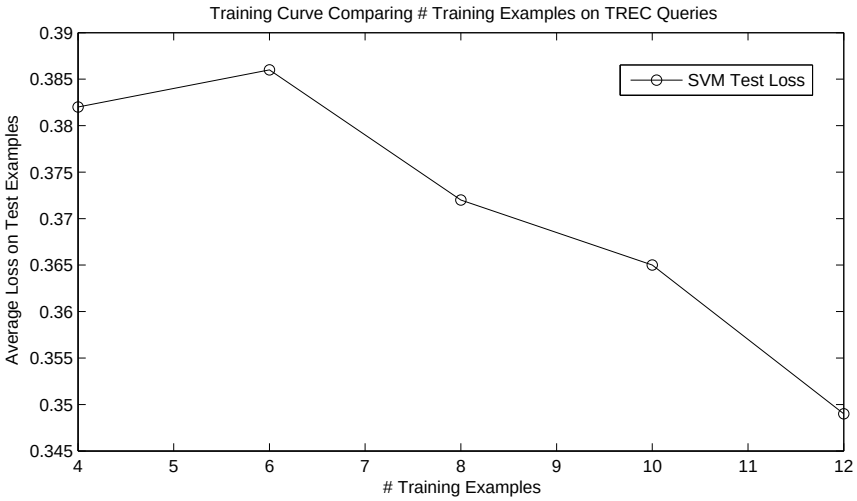

Figure 4.5: Diversified Retrieval: Comparing Training Size on TREC Dataset ($K = 5$)

Figure 4.5 shows the average test performance of $\text{SVM}^{\Delta}_{div}$ as the number of training examples is varied. We see a substantial improvement in performance as training set size increases. It appears that more training data would further improve performance.

4.4.2 Approximate Constraint Generation

Using greedy constraint generation might cause the learning aproach to underfit the data. A simple way to check for underfitting is to examine training loss as the C parameter is varied. The training curve of $\text{SVM}^{\Delta}_{div}$ is shown in Figure 4.6. Greedy optimal refers to the loss incurred by a greedy method with knowledge of subtopics. As we increase C (favoring low training loss over low model complexity), the model is able to fit the training data almost as well as the clairvoyant greedy approach. This indicates that using the greedy method for constraint generation is acceptable for for this learning task.

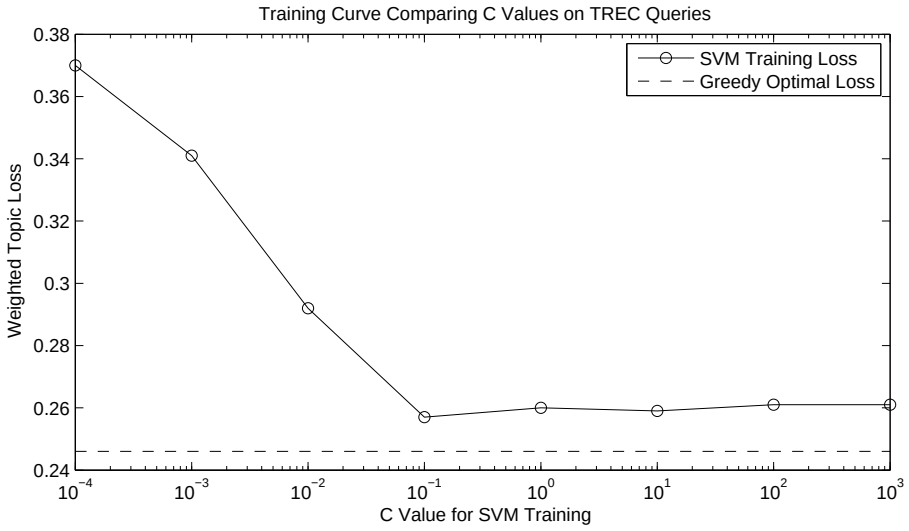

Figure 4.6: Diversified Retrieval: Comparing C Values on TREC Dataset ($K = 5$)

4.4.3 Varying Predicted Subset Size

We used the synthetic dataset to evaluate performance as we vary the retrieval size K. It is difficult to perform this evaluation on the TREC queries – since some candidate sets have very few documents or subtopics, using higher K would force us to discard more queries. Figure 4.7 shows that the test performance of $\text{SVM}^{\Delta}_{div}$ consistently outperforms Essential Pages at all levels of K.

4.5 Extensions

4.5.1 Alternative Discriminants

Maximizing word coverage might not be suitable for other types of retrieval tasks. Our method is a general framework which can incorporate other discriminant formulations. One possible alternative is to maximize the pairwise distance of items

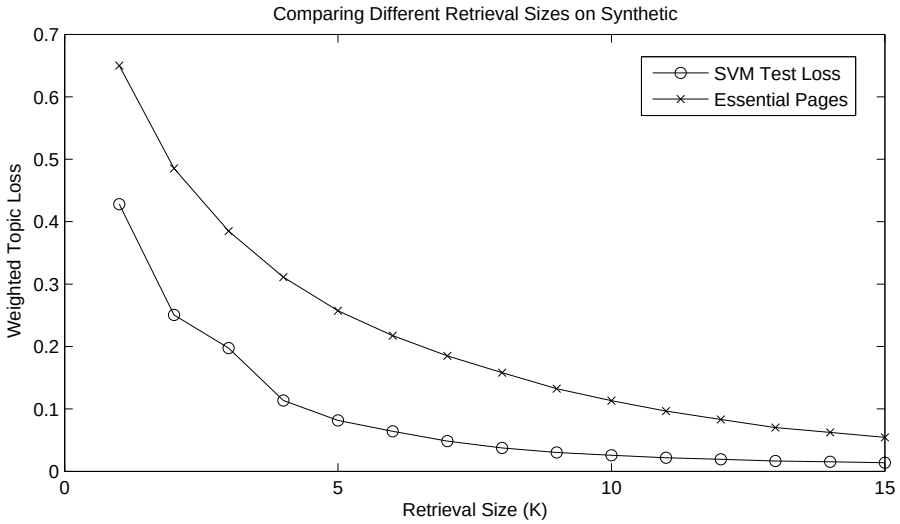

Figure 4.7: Diversified Retrieval: Varying Retrieval Size on Synthetic Dataset

in the predicted subset. Learning a weight vector for (3.1) would then amount to finding a distance function for a specific retrieval task. Any discriminant can be used so long as it captures the salient properties of the retrieval task, is linear in a joint feature space (3.1), and has effective inference and constraint generation methods.

4.5.2 Alternative Loss Functions

Our method is not restricted to using subtopics to measure diversity. Only the loss function $\Delta(\mathbb{T}, \mathbf{y})$ makes use of subtopics during SVM training. We can also incorporate loss functions which can penalize other types of diversity criteria and also use other forms of training data, such as clickthrough logs. The only requirement is that it must be computationally compatible with the constraint generation oracle (4.6).

4.5.3 Additional Word Features

Our choice of features is based almost exclusively on word frequencies. The sole exception is using title words as an importance criterion. The goal of these features is to describe how well a document covers a word and the importance of covering a word in a candidate set. Other types of word features might prove useful, such as anchor text, URL, and any meta information contained in the documents.

4.6 Discussion

Diversified retrieval is not the only task that can be viewed as a coverage problem. For example, consider the document summarization task [52]. Given a small collection of documents, the goal is to develop approaches to automatically extract a subset of sentences that best summarizes the collection. A common evaluation measure is the ROUGE score [115]. Given gold-standard sentences extracted by human judges y, the ROUGE score of a prediction p essentially computes the degree to which p covers y (e.g., how many words contained in y are also contained in p). Unsurprisingly, many researchers have proposed approaches that optimize for coverage models that do not rely on human annotations (cf. [116]). The $\text{SVM}^{\Delta}_{div}$ approach can be naturally applied here to train compex models containing many parameters with the explicit goal of optimizing ROUGE score on a training set.

In some sense, structured prediction learning is the "inverse" of solving combinatorial optimization problems. For example, when given detailed information of the true objective function (e.g., having access to the subtopic labels), one can ex-

ploit combinatorial structure (e.g., submodularity) in order to find a good solution. The inverse problem is to find the best possible surrogate model that can be used in more general settings (such when ground truth labels are unavailable). From a machine learning perspective, this task can be restated as training the parameters of a general parameterized model (which ideally has a combinatorial structure that is compatible with the true loss/objective function) such that the predictions of the parameterized model (i.e., the solution to solving the resulting combinatorial optimization problem induced by the model) matches the solution to the original problem (i.e., has low loss).

One limitation of many existing structured prediction approaches is the need for labeled data. It is typically much easier to define conceptually satisfying utiliy functions (or loss functions) for training when given labeled data, which can be difficult to acquire. Moreover, in many structured prediction tasks, supervision may only be available at a coarse level, leading to a latent variable learning problem [178, 36]. This limitation is a major motivation for a complementary line of research on interactive learning (i.e., a machine learning algorithm that can collect feedback from interacting with users) to be presented in the following chapters.

4.6.1 Beyond Predicting Static Rankings

It is common for information retrieval research to focus either on relevance estimation or user interface design, but rarely both simultaneously. However, for many tasks, it can be useful to model both jointly. For instance, as potentially relevant information become more heterogeneous (possibly due to ambiguity in estimating users' interests), then it may prove more beneficial to move away from displaying

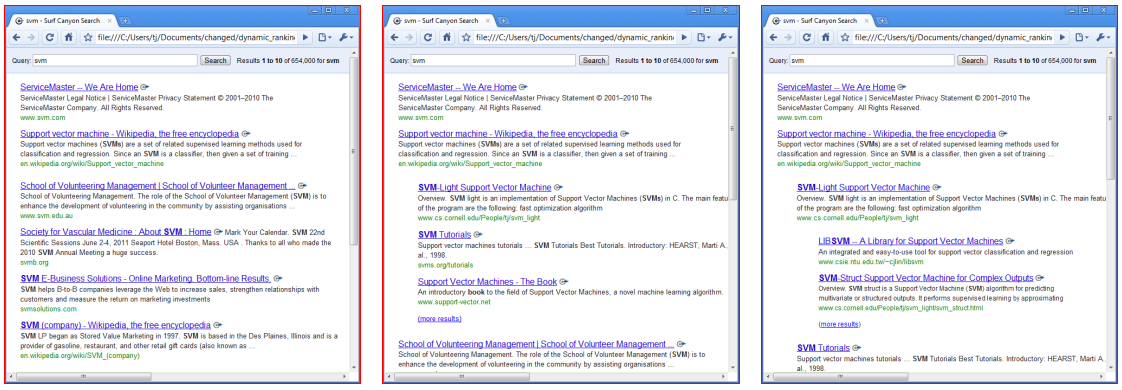

Figure 4.8: Example of user interacting with dynamic ranking.

results using static, one-dimensional rankings and towards richer layouts.[3] One major limitation of result diversification over static rankings is that it sacrifices recall in favor of some minimal amount of utility for all usage intents – such a limitation could be dealt with by moving towards more dynamic interfaces.

Consider the example interface shown in Figure 4.8, which is inspired by and adapted from the SurfCanyon.com search engine [48]. In this example, the user first receives a conventional diversified ranking in response to the query SVM (Figure 4.8, left). However, by clicking or mousing over a result that matches the users intent, additional indented results are inserted into the original ranking1 (Figure 4.8, middle). This process can be repeated multiple levels deep (Figure 4.8, right). This interaction is quite natural, since the process resembles navigating a drop-down menu and since users are already familiar with result indentation. And yet even this one additional degree of freedom in content display can offer tremendous benefits. In the example, the indented results have greatly improved recall for the users information need on the learning method Support Vector Machine. While there is only a single relevant document in the original ranking, the final ranking

[3] In fact, even standard search services present other potentially relevant information in addition to the retrieved results, with the most notable being web advertisements. But virtually all research have focused on one of the two aspects in isolation.

covers many aspects of the learning method.

The interplay between content and interface becomes more pronounced as one considers settings that deal with increasingly heterogeneous content. Consider the setting of optimizing the information content of an internet based vendor (e.g., Amazon, Netflix) or a media service (e.g., Hulu, YouTube). These services offer information that can be customized both to particular users as well as to varying types information requests. Furthermore, the retrieved content is rarely presented in a clean, one-dimensional ranking. Structured prediction approaches offer the potential to develop rich models that can jointly quantify retrieval quality over both content relevance and placement (i.e. interface design) that can be optimized using domain-specific training data.

Part III

Interactive Learning

CHAPTER 5

INTERACTIVE LEARNING AND THE DUELING BANDITS PROBLEM

When responding to queries, the goal of an information retrieval system – ranging from web search, to desktop search, to call center support – is to return the results that maximize user utility. The conventional approach – which we adopted for the methods described in Chapter 3 – is to optimize a proxy-measure that is hoped to correlate with utility. A wide range of measures has been proposed to this effect (e.g., average precision, precision at k, NDCG), but all have similar problems. Most notably, they require expensive manual relevance judgments that ignore the identity of the user and the user's context.[1] This makes it unclear whether maximization of a proxy-measure truly optimizes the search experience for the user.

In this chapter, we therefore take a different approach based on interacting with and gathering implicit feedback directly from users. But how can a learning algorithm access the utility a user sees in a set of results? While it is unclear how to reliably derive cardinal utility values for a set of results (e.g., $U(r) = 5.6$), it was shown that interactive experiments can reliably provide ordinal judgments between two sets of results (i.e., $U(r_1) > U(r_2)$) [88, 140]. For example, to elicit whether a user prefers ranking r_1 over r_2, Radlinski et al. [140] showed how to present an interleaved ranking of r_1 and r_2 so that clicks indicate which of the two has higher utility.[2] This ready availability of pairwise comparison feedback in

[1] Although beyond the scope of this dissertation, it should be noted that these problems become even more pronounced when attempting to define proxy-measures to characterize the diversified retrieval setting described in Chapter 4.

[2] The interleaving mechanism is described in greater detail in Chapter 7. For this chapter, we simply assume the availability of an unbiased comparison oracle for the target application domain, which in our case is search.

applications where absolute payoffs are difficult to observe motivates our learning framework.

Algorithms which can perform well in this setting must *interact* with users by pro-actively choosing which results to show them (e.g., which two retrieved rankings to interleave). In particular, they must choose which retrieval functions to compare in order to optimize for a model of user utility described in the following section – this utility function leads to a natural and well-founded trade-off between exploration and exploitation. On one hand, if we only passively collect feedback from the incumbent ranking function (as is often done in practice), then we run the risk of never discovering the best retrieval function – this point was also touched on in Section 2.5. On the other hand, if we only explore (e.g., by always comparing new pairs of retrieval functions), then we might be using suboptimal retrieval functions for far longer than we need to; this leads to suboptimal performance since user utility accumulates with every comparison.

For the rest of this chapter, we proceed by first introducing the Dueling Bandits Problem [180, 179, 185] for modeling such interactive learning scenarios. We then present efficient algorithms for the discrete setting (where there are K retrieval functions to choose from), and prove performance guarantees [180, 179]. We conclude by discussing of related work and presenting empirical simulation results. Chapter 6 describes a method for optimizing the Dueling Bandits Problem in the continuous setting (where we must choose from a continuously parameterized family of retrieval functions).

5.1 The K-armed Dueling Bandits Problem

We propose a new online optimization problem, called the Dueling Bandits Problem, where the only actions are comparisons (or duels) between two bandits within a space of bandits $\mathcal{B}$ (e.g., a collection of candidate retrieval functions for a search engine). We assume that the outcomes of these noisy comparisons are independent random variables[3] and that the probability of bandit b winning a comparison with bandit b' is stationary over time. We write this probability as $P(b > b') = \epsilon(b, b') + 1/2$, where $\epsilon(b, b') \in (-1/2, 1/2)$ is a measure of the distinguishability between b and b'. We assume that there exists a total ordering on $\mathcal{B}$ such that $b \succ b'$ implies $\epsilon(b, b') > 0$. We will also use the notation $\epsilon_{i,j} \equiv \epsilon(b_i, b_j)$.

We quantify the performance of an online algorithm using the following regret formulations. Let $(b_1^{(t)}, b_2^{(t)})$ be the bandits chosen at iteration t, and let b^* be the overall best bandit. We define **strong regret** based on comparing the chosen bandits with b^*,

$$R_T = \frac{1}{2} \sum_{t=1}^{T} \left(\epsilon(b^*, b_1^{(t)}) + \epsilon(b^*, b_2^{(t)}) \right), \tag{5.1}$$

where T is the time horizon. We also define **weak regret**,

$$\tilde{R}_T = \sum_{t=1}^{T} \min\{\epsilon(b^*, b_1^{(t)}), \epsilon(b^*, b_2^{(t)})\}, \tag{5.2}$$

which only compares $\hat{b}$ against the better of $b_1^{(t)}$ and $b_2^{(t)}$. One can regard regret as essentially the fraction of users who would have preferred the best bandit over the chosen ones in each iteration.[4] More precisely, it corresponds to the fraction

[3] For example, the probabiltiy of b_i winning a comparison with b_j in any given comparison can be sampled from a Bernoulli distribution μ_{ij}. The independence assumption here requires that the result of any particular comparison, conditioned on the pair of bandits b_i and b_j (thus μ_{ij}), is sampled independently of all other comparisons between any pair of bandits.

[4] In the search setting, users experience an interleaving, or mixing, of results from both retrieval functions to be compared.

of users who prefer the best bandit to a uniformly-random member of the pair of bandits chosen, in the case of strong regret, or to the better of the two bandits chosen, in the case of weak regret. Building from this perspective, we can also define **generalized regret**,

$$\bar{R}_T = \sum_{t=1}^{T} r_t(b_1^{(t)}, b_2^{(t)}), \tag{5.3}$$

where

$$r_t(b_1^{(t)}, b_2^{(t)}) \in \left[\min\{\epsilon(b^*, b_1^{(t)}), \epsilon(b^*, b_2^{(t)})\},\ \max\{\epsilon(b^*, b_1^{(t)}), \epsilon(b^*, b_2^{(t)})\}\right].$$

At each time step t, $r_t(b_1^{(t)}, b_2^{(t)})$ is the (potentially non-deterministic) incurred regret of comparing $b_1^{(t)}$ and $b_2^{(t)}$ and is assumed to be bounded between the two individual regret values. Note that both strong regret and weak regret are special cases where $r_t(b_1^{(t)}, b_2^{(t)}) = (\epsilon(b^*, b_1^{(t)}) + \epsilon(b^*, b_2^{(t)}))/2$ and $r_t(b_1^{(t)}, b_2^{(t)}) = \min\{\epsilon(b^*, b_1^{(t)}), \epsilon(b^*, b_2^{(t)})\}$, respectively. We will present algorithms which achieve identical regret bounds for all three formulations (up to constant factors) by assuming a property called stochastic triangle inequality, which is described in the next section.

In this chapter, we assume a discrete space of K bandits, i.e. $\mathcal{B} = \{b_1, \ldots, b_K\}$; we call this setting the K-armed Dueling Bandits Problem. We will consider a continuum-armed instance of the Dueling Bandits Problem in Chapter 6.

5.2 Modeling Assumptions

We impose additional structure to the probabilistic comparisons. First, we assume **strong stochastic transitivity**, which requires that any triplet of bandits $b_i \succ b_j \succ b_k$ satisfies

$$\epsilon_{i,k} \geq \max\{\epsilon_{i,j}, \epsilon_{j,k}\}. \tag{5.4}$$

This assumption provides a monotonicity constraint on possible probability values.

We also assume **stochastic triangle inequality**, which requires any triplet of bandits $b_i \succ b_j \succ b_k$ to satisfy

$$\epsilon_{i,k} \leq \epsilon_{i,j} + \epsilon_{j,k}. \tag{5.5}$$

Stochastic triangle inequality captures the condition that the probability of different bandits winning (or losing) a comparison will exhibit diminishing returns as they become increasingly superior (or inferior) to the competing bandit.[5]

We briefly describe two common generative models which satisfy these two assumptions. The first is the logistic or Bradley-Terry model, where each bandit b_i is assigned a positive real value μ_i. Probabilistic comparisons are made using

$$P(b_i > b_j) = \frac{\mu_i}{\mu_i + \mu_j}.$$

The second is a Gaussian model, where each bandit is associated with a random variable X_i that has a Gaussian distribution with mean μ_i and variance 1. Probabilistic comparisons are made using

$$P(b_i > b_j) = P(X_i - X_j > 0),$$

where $X_i - X_j \sim N(\mu_i - \mu_j, 2)$. It is straightforward to check that both models satisfy strong stochastic transitivity and stochastic triangle inequality. We will describe and justify a more general family of probabilistic models in Appendix B.1.

[5]Our analysis also applies for a relaxed version where $\epsilon_{i,k} \leq \gamma(\epsilon_{i,j} + \epsilon_{j,k})$ for finite $\gamma > 0$.

Algorithm 4 Explore Then Exploit Solution

1: Input: T, $\mathcal{B} = \{b_1, \ldots, b_K\}$, $EXPLORE$
2: $(\hat{b}, \hat{T}) \leftarrow EXPLORE(T, \mathcal{B})$
3: **for** $t = \hat{T} + 1, \ldots, T$ **do**
4: compare $\hat{b}$ and $\hat{b}$
5: **end for**

5.3 Algorithm and Analysis

Our solution, which is described in Algorithm 4, follows an "explore then exploit" approach. For a given time horizon T and a set of K bandits $\mathcal{B} = \{b_1, \ldots, b_K\}$, an exploration algorithm (denoted generically as EXPLORE) is used to find the best bandit b^*. EXPLORE returns both its solution $\hat{b}$ as well as the total number of iterations $\hat{T}$ for which it ran (it is possible that $\hat{T} > T$). Should $\hat{T} < T$, we enter an exploit phase by repeatedly choosing $(b_1^{(t)}, b_2^{(t)}) = (\hat{b}, \hat{b})$, which incurs no additional regret assuming EXPLORE correctly found the best bandit ($\hat{b} = b^*$). In the case where $\hat{T} > T$, then the regret incurred from running EXPLORE still bounds our regret formulations (which only measures regret up to T), so our analysis in this section will still hold.[6]

We will consider two versions of our proposed exploration algorithm, which we call Interleaved Filter 1 (IF1) and Interleaved Filter 2 (IF2). We will show that both algorithms (which we refer to generically as IF) correctly return the best bandit with probability at least $1 - 1/T$. Correspondingly, a suboptimal bandit is returned with probability at most $1/T$, in which case we assume maximal regret

[6] In practice, we can terminate EXPLORE after it has run for T time steps, in which case the incurred regret is strictly less than running EXPLORE to completion.

$\mathcal{O}(T)$. We can thus bound the expected regret by

$$\begin{aligned}\mathbf{E}[R_T] &\leq \left(1 - \frac{1}{T}\right) \mathbf{E}\left[R_T^{IF}\right] + \frac{1}{T}\mathcal{O}(T) \\ &= \mathcal{O}\left(\mathbf{E}\left[R_T^{IF}\right] + 1\right) \end{aligned} \tag{5.6}$$

where R_T^{IF} denotes the regret incurred from running Interleaved Filter. Thus the regret bound depends entirely on the regret incurred by Interleaved Filter.

The two IF algorithms are described in Algorithm 5 and Algorithm 6, respectively. IF2 achieves an expected regret bound which matches the information-theoretic lower bound (up to constant factors) presented in Section 5.3.5, whereas IF1 matches with high probability the lower bound up to a log factor. We first examine IF1 due to its ease of analysis. We then analyze IF2, which builds upon IF1 to achieve the information-theoretic optimum.

In both versions, IF maintains a candidate bandit $\hat{b}$ and simulates simultaneously comparing $\hat{b}$ with all other remaining bandits via round robin scheduling (i.e., interleaving). Any bandit that is empirically inferior to $\hat{b}$ with $1 - \delta$ confidence is removed (we will describe later how to choose δ). When some bandit b' is empirically superior to $\hat{b}$ with $1 - \delta$ confidence, then $\hat{b}$ is removed and b' becomes the new candidate $\hat{b} \leftarrow b'$. IF2 contains an additional step where all empirically inferior bandits (even if lacking $1-\delta$ confidence) are removed (called pruning – see lines 16-18 in Algorithm 6). This process repeats until only one bandit remains. Assuming IF has not made any mistakes, then it will return the best bandit $\hat{b} = b^*$.

Terminology. Interleaved Filter makes a "**mistake**" if it draws a false conclusion regarding a pair of bandits. A mistake occurs when an inferior bandit is determined with $1 - \delta$ confidence to be the superior one. We call the additional step of IF2 (lines 16-18 in Algorithm 6) "**pruning**". We define a "**match**" to be

all the comparisons Interleaved Filter makes between two bandits, and a "**round**" to be all the matches played by one candidate $\hat{b}$. We always refer to $\log x$ as the natural log, $\ln x$, whenever the distinction is necessary.

In our analysis, we assume without loss of generality that the bandits in $\mathcal{B}$ are sorted in preferential order $b_1 \succ \ldots \succ b_K$. Then for $T \geq K$, we will show in Theorem 3 that running IF1 incurs, with high probability, regret bounded by

$$R_T^{IF1} = \mathcal{O}\left(\frac{K \log K}{\epsilon_{1,2}} \log T\right).$$

Note that $\epsilon_{1,2} = P(b_1 \succ b_2) - 1/2$ is the distinguishability between the two best bandits. Due to strong stochastic transitivity, $\epsilon_{1,2}$ lower bounds the distinguishability between the best bandit and any other bandit. We will also show in Theorem 4 that running IF2 incurs expected regret bounded by

$$\mathbf{E}\left[R_T^{IF2}\right] = \mathcal{O}\left(\frac{K}{\epsilon_{1,2}} \log T\right),$$

which matches the information-theoretic lower bound (up to constant factors) described in Section 5.3.5.

Analysis Approach. Our analysis follows three phases. We first bound the regret incurred for any match. Then for both IF1 and IF2, we show that the probability of making a mistake[7] is at most $1/T$. We finally bound the number of matches played by IF1 and IF2 to arrive at our final regret bounds.

[7]This is the probability that our algorithms come to the wrong conclusion regarding any pair of bandits. Thus, our analysis is conservative since our algorithms can potentially recover from making a few mistakes.

Algorithm 5 Interleaved Filter 1 (IF1)

Input: T, $\mathcal{B} = \{b_1, \ldots, b_K\}$
$\delta \leftarrow 1/(TK^2)$
Choose $\hat{b} \in \mathcal{B}$ randomly
$W \leftarrow \{b_1, \ldots, b_K\} \setminus \{\hat{b}\}$
$\forall b \in W$, maintain estimate $\hat{P}_{\hat{b},b}$ of $P(\hat{b} > b)$ according to (5.7)
$\forall b \in W$, maintain $1-\delta$ confidence interval $\hat{C}_{\hat{b},b}$ of $\hat{P}_{\hat{b},b}$ according to (5.8), (5.9)
while $W \neq \emptyset$ **do**
  for $b \in W$ **do**
    compare $\hat{b}$ and b
    update $\hat{P}_{\hat{b},b}$, $\hat{C}_{\hat{b},b}$
  end for
  while $\exists b \in W$ s.t. $\left(\hat{P}_{\hat{b},b} > 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b}\right)$ **do**
    $W \leftarrow W \setminus \{b\}$ *//$\hat{b}$ declared winner against b*
  end while
  if $\exists b' \in W$ s.t. $\left(\hat{P}_{\hat{b},b'} < 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b'}\right)$ **then**
    $\hat{b} \leftarrow b'$, $W \leftarrow W \setminus \{b'\}$ *//b' declared winner against $\hat{b}$ (new round)*
    $\forall b \in W$, reset $\hat{P}_{\hat{b},b}$ and $\hat{C}_{\hat{b},b}$
  end if
end while
$\hat{T} \leftarrow$ Total Comparisons Made
return $(\hat{b}, \hat{T})$

5.3.1 Confidence Intervals

In a match between b_i and b_j, Interleaved Filter maintains a number

$$\hat{P}_{i,j} = \frac{\#\ b_i\ wins}{\#\ comparisons\ b_i\ vs\ b_j}, \tag{5.7}$$

which is the empirical estimate of $P(b_i \succ b_j)$ after t comparisons.[8] For ease of notation, we drop the subscripts (b_i, b_j), and use $\hat{P}_t$, which emphasizes the dependence on the number of comparisons. IF also maintains a confidence interval

$$\hat{C}_t = (\hat{P}_t - c_t, \hat{P}_t + c_t), \tag{5.8}$$

where

$$c_t = \sqrt{4 \log(1/\delta)/t}. \tag{5.9}$$

[8] In other words, $\hat{P}_{i,j}$ is the fraction of these t comparisons in which b_i was the winner.

We justify the construction of these confidence intervals in the following lemma.

Lemma 2. *For $\delta = 1/(TK^2)$, the number of comparisons in a match between b_i and b_j is with high probability at most*

$$\mathcal{O}\left(\frac{1}{\epsilon_{i,j}^2}\log(TK)\right).$$

Moreover, the probability that the inferior bandit is declared the winner at some time $t \leq T$ is at most δ.

Proof. First we argue that the probability of the inferior bandit being declared the winner is at most δ. Note that by the stopping condition of the match, if we mistakenly declare the inferior bandit the winner at time t, then we must have $1/2 + \epsilon_{i,j} \notin \hat{C}_t$ (note that $\epsilon_{i,j}$ can be either positive or negative). By the definition of $\hat{C}_t$ and the fact that $\mathbf{E}[\hat{P}_t] = 1/2 + \epsilon_{i,j}$, we have $P(1/2 + \epsilon_{i,j} \notin \hat{C}_t) = P(|\hat{P}_t - \mathbf{E}[\hat{P}_t]| \geq c_t)$. It follows from Hoeffding's inequality [80] that the probability of making a mistake at time t is bounded above by

$$P(|\hat{P}_t - \mathbf{E}[\hat{P}_t]| \geq c_t) \leq 2\exp(-2tc_t^2) = 2\exp(-8\log(1/\delta)) = 2\delta^8 = \frac{2}{T^8K^{16}}.$$

Now an application of the union bound shows that the probability of making a mistake at any time $t \leq T$ is bounded above by

$$P\left(\bigcup_{t=1}^{T}\{1/2 + \epsilon_{i,j} \notin \hat{C}_t\}\right) \leq \frac{2T}{T^8K^{16}} \leq \frac{1}{TK^2} = \delta,$$

provided that $K \geq 2$, which is the desired result.

We now show that the number of comparisons n in a match between b_i and b_j is $\mathcal{O}(\log(TK)/\epsilon_{i,j}^2)$ with high probability. Specifically, we will show that for any $d \geq 1$, there exists an m depending only on d such that

$$P\left(n \geq \frac{m}{\epsilon_{i,j}^2}\log(TK)\right) \leq K^{-d}$$

for all K sufficiently large. By the stopping condition of the match, if at any time t we have $\hat{P}_t - c_t > 1/2$, then the match terminates. It follows that for any time t, if $n > t$, then $\hat{P}_t - c_t \leq 1/2$, and so

$$P(n > t) \leq P(\hat{P}_t - c_t \leq 1/2).$$

To bound this probability, assume without loss of generality that $\epsilon_{i,j} > 0$, and note that since $\mathbf{E}[\hat{P}_t] = 1/2 + \epsilon_{i,j}$, we have

$$P(\hat{P}_t - c_t \leq 1/2) = P(\hat{P}_t - 1/2 - \epsilon_{i,j} \leq c_t - \epsilon_{i,j}) = P(\mathbf{E}[\hat{P}_t] - \hat{P}_t \geq \epsilon_{i,j} - c_t).$$

For any $m \geq 8$ and $t \geq \lceil 2m \log(TK^2)/\epsilon_{i,j}^2 \rceil$, we have $c_t \leq \epsilon_{i,j}/2$, and so applying Hoeffding's inequality for this m and t shows

$$P(\mathbf{E}[\hat{P}_t] - \hat{P}_t \geq \epsilon_{i,j} - c_t) \leq P(|\hat{P}_t - \mathbf{E}[\hat{P}_t]| \geq \epsilon_{i,j}/2) \leq 2\exp(-t\epsilon_{i,j}^2/2).$$

Since $t \geq 2m \log(TK^2)/\epsilon_{i,j}^2$ by assumption, we have $t\epsilon_{i,j}^2/2 \geq m \log(TK^2)$, and so

$$2\exp(-t\epsilon_{i,j}^2/2) \leq 2\exp(-m \log(TK^2)) = \frac{2}{T^m K^{2m}} \leq K^{-m}$$

for $K \geq 2$, which proves the claim.

□

5.3.2 Regret per Match

We now bound the accumulated regret of each match. We first bound strong and weak regret, and then extend the result to generalized regret.

Lemma 3. *Assuming b_1 has not been removed and $T \geq K$, then with high probability the accumulated weak regret and also strong regret from any match is at most*

$$\mathcal{O}\left(\frac{1}{\epsilon_{1,2}} \log T\right).$$

Proof. Suppose the candidate bandit $\hat{b} = b_j$ is playing a match against b_i. Since all matches within a round are played simultaneously, then by Lemma 2, any match played by b_j contains at most

$$\mathcal{O}\left(\frac{1}{\epsilon_{1,j}^2}\log(TK)\right) \leq \mathcal{O}\left(\frac{1}{\epsilon_{1,2}^2}\log(TK)\right)$$

comparisons, where the inequality follows from strong stochastic transitivity. Note that $\min\{\epsilon_{1,j}, \epsilon_{1,i}\} \leq \epsilon_{1,j}$. Then the accumulated weak regret (5.2) is bounded by

$$\begin{aligned}
\epsilon_{1,j}\mathcal{O}\left(\frac{1}{\epsilon_{1,j}^2}\log(TK)\right) &= \mathcal{O}\left(\frac{1}{\epsilon_{1,j}}\log(TK)\right) \\
&\leq \mathcal{O}\left(\frac{1}{\epsilon_{1,2}}\log(TK)\right) \\
&= \mathcal{O}\left(\frac{1}{\epsilon_{1,2}}\log T\right) \qquad (5.10)
\end{aligned}$$

where (5.10) holds since $\log(TK) \leq \log(T^2) = 2\log T$. We now bound the accumulated strong regret (5.1) by leveraging stochastic triangle inequality. Each comparison incurs $\epsilon_{1,j} + \epsilon_{1,i}$ regret. We consider the following three cases.

Case 1: Suppose $b_i \succ b_j$. Then $\epsilon_{1,j} + \epsilon_{1,i} \leq 2\epsilon_{1,j}$, and the accumulated strong regret of the match is bounded by

$$2\epsilon_{1,j}\mathcal{O}\left(\frac{1}{\epsilon_{1,j}^2}\log(TK)\right) \leq \mathcal{O}\left(\tfrac{1}{\epsilon_{1,2}}\log(TK)\right)$$

Case 2: Suppose $b_j \succ b_i$ and $\epsilon_{j,i} \leq \epsilon_{1,j}$. Then

$$\begin{aligned}
\epsilon_{1,j} + \epsilon_{1,i} &\leq \epsilon_{1,j} + \epsilon_{1,j} + \epsilon_{j,i} \\
&\leq 3\epsilon_{1,j}
\end{aligned}$$

and the accumulated strong regret is bounded by

$$\begin{aligned}
3\epsilon_{1,j}\mathcal{O}\left(\frac{1}{\epsilon_{1,j}^2}\log(TK)\right) &= \mathcal{O}\left(\tfrac{1}{\epsilon_{1,j}}\log(TK)\right) \\
&\leq \mathcal{O}\left(\tfrac{1}{\epsilon_{1,2}}\log(TK)\right)
\end{aligned}$$

Case 3: Suppose $b_j \succ b_i$ and $\epsilon_{j,i} > \epsilon_{1,j}$. Then we can also use Lemma 2 to bound with high probability the number of comparisons by

$$\mathcal{O}\left(\frac{1}{\epsilon_{j,i}^2}\log(TK)\right).$$

The accumulated strong regret is then bounded by

$$\begin{aligned} 3\epsilon_{j,i}\mathcal{O}\left(\frac{1}{\epsilon_{j,i}^2}\log(TK)\right) &= \mathcal{O}\left(\tfrac{1}{\epsilon_{j,i}}\log(TK)\right) \\ &\leq \mathcal{O}\left(\tfrac{1}{\epsilon_{1,j}}\log(TK)\right) \\ &\leq \mathcal{O}\left(\tfrac{1}{\epsilon_{1,2}}\log(TK)\right) \end{aligned}$$

Like in the analysis for weak regret (5.10), we finally note that

$$\mathcal{O}\left(\frac{1}{\epsilon_{1,2}}\log(TK)\right) = \mathcal{O}\left(\frac{1}{\epsilon_{1,2}}\log T\right).$$

□

Lemma 4. *Assuming b_1 has not been removed and $T \geq K$, then with high probability the accumulated generalized regret from any match is at most*

$$\mathcal{O}\left(\frac{1}{\epsilon_{1,2}}\log T\right).$$

Proof. Suppose the candidate bandit $\hat{b} = b_j$ is playing a match against b_i. At each time step t that b_i is compared to b_j, the accumulated generalized regret for that comparison is $r(b_i, b_j) \in [\min\{\epsilon_{1,i}, \epsilon_{1,j}\}, \max\{\epsilon_{1,i}, \epsilon_{1,j}\}]$. Let n denote the number of comparisons made in the match. Then the accumulated generalized regret can be bounded by

$$n\max\{\epsilon_{1,i}, \epsilon_{1,j}\} \leq n\left(\epsilon_{1,i} + \epsilon_{1,j}\right) = \mathcal{O}\left(\frac{1}{\epsilon_{1,2}}\log T\right),$$

where the last equality is the regret bound for strong regret derived in Lemma 3. □

In the next two sections, we will bound the mistake probability and total matches played by IF1 and IF2, respectively.

5.3.3 Regret Bound for Interleaved Filter 1

We first state our main regret bound for Interleaved Filter 1.

Theorem 3. *Running Algorithm 4 with $\mathcal{B} = \{b_1, \ldots, b_K\}$, time horizon T ($T \geq K$), and IF1 incurs expected generalized regret (and thus also weak and strong regret) bounded by*

$$\mathbf{E}[R_T] \leq \mathcal{O}\left(\mathbf{E}\left[R_T^{IF1}\right]\right) = \mathcal{O}\left(\frac{K \log K}{\epsilon_{1,2}} \log T\right).$$

The theorem will follow from combining Lemma 4, (5.6), and Lemmas 5 and 7 to follow. We begin by analyzing the probability of IF1 making a mistake.

Lemma 5. *IF1 makes a mistake with probability at most $1/T$.*

Proof. By Lemma 2, the probability that IF1 makes a mistake in any given match is at most $1/(TK^2)$. Since K^2 is a trivial upper bound on the number of matches, applying the union bound over all matches proves the lemma. □

We assume for the remainder of this section that IF1 is mistake-free, since the cost of making a mistake is considered in (5.6), and we are interested here in bounding R_T^{IF1}. We can model the sequence of candidate bandits using the following random walk model.

Definition 1. ***(Random Walk Model)*** *Define a random walk graph with K nodes labeled $b_1, \ldots, b_K$ (these will correspond to the similarly named bandits).*

Each node b_j ($j > 1$) transitions to b_i for $j > i \geq 1$ with probability $1/(j-1)$, or in other words b_j transitions to $b_1, \ldots, b_{j-1}$ with uniform probability. The final node b_1 is an absorbing node.

A path in the Random Walk Model corresponds to a sequence of candidate bandits taken by IF (both IF1 and IF2) in an instance of the Dueling Bandits problem where $\epsilon_{1j} = \epsilon_{2j} = \ldots = \epsilon_{j-1,j}$ for all $j > 1$ (and no mistakes are made). Thus, the path length of the random walk is exactly to the number of rounds in IF.

Proposition 2. *Either IF makes a mistake, or else the number of rounds in the execution of IF is stochastically dominated by the path length of a random walk in the Random Walk Model.*

Proposition 2 follows directly from Lemma 21 in Appendix B.2. This allows us to concentrate our analysis on the (simpler) upper bound setting of the Random Walk Model. We will prove that the random walk in the Random Walk Model requires $\mathcal{O}(\log K)$ steps with high probability. Let X_i ($1 \leq i < K$) be an indicator random variable corresponding to whether a random walk starting at b_K visits b_i in the Random Walk Model. We first analyze the marginal probability of each $P(X_i = 1)$, and also show that $X_1, \ldots, X_{K-1}$ are mutually independent.

Lemma 6. *Let X_i be as defined above with $1 \leq i < K$. Then*

$$P(X_i = 1) = \frac{1}{i},$$

and furthermore, for all $W \subseteq \{X_1, \ldots, X_{K-1}\}$, we can write $P(W) \equiv P(\bigwedge_{i \in W} X_i)$ as

$$P(W) = \prod_{X_i \in W} P(X_i), \tag{5.11}$$

meaning $X_1, \ldots, X_{K-1}$ are mutually independent.

Proof. We can rewrite (5.11) as

$$P(W) = \prod_{X_i \in W} P(X_i|W_i),$$

where $W_i = \{X_j \in W | j > i\}$.

We first consider $W = \{X_1, \ldots, X_{K-1}\}$. For the factor on X_i, denote with j the smallest index in W_i with $X_j = 1$ in the condition. Then

$$\begin{aligned} &P(X_i = 1|X_{i+1}, ..., X_{K-1}) \\ &\quad = P(X_i = 1|X_{i+1} = 0, ..., X_{j-1} = 0, X_j = 1) = \frac{1}{i}, \end{aligned}$$

since the walk moved to one of the first i nodes with uniform probability independent of j. Since $\forall j > i : P(X_i = 1|X_j = 1) = \frac{1}{i}$, this implies $P(X_i = 1) = \frac{1}{i}$. So we can conclude

$$P(X_1, \ldots, X_{K-1}) = \prod_{i=1}^{K-1} P(X_i).$$

Now consider arbitrary W. We use $\sum_{W^c}$ to indicate summing over the joint states of all X_i variables not in W. We can write $P(W)$ as

$$\begin{aligned} P(W) &= \sum_{W^c} P(X_1, \ldots, X_{K-1}) \\ &= \sum_{W^c} \prod_{i=1}^{K-1} P(X_i) \\ &= \prod_{X_i \in W} P(X_i) \left(\sum_{W^c} \prod_{X_i \in W^c} P(X_i) \right) \\ &= \prod_{X_i \in W} P(X_i). \end{aligned}$$

This proves mutual independence (5.11).

□

We can express the number of steps taken by a random walk from b_K to b_1 in the Random Walk Model as

$$S_K = 1 + \sum_{i=1}^{K-1} X_i. \tag{5.12}$$

Lemma 6 implies that

$$E[S_K] = 1 + \sum_{i=1}^{K-1} E[X_i] = 1 + H_{K-1} \approx \log K,$$

where H_i is the harmonic sum. We now show that $S_K = \mathcal{O}(\log K)$ with high probability.

Lemma 7. *Assuming IF1 is mistake-free, then it runs for $\mathcal{O}(\log K)$ rounds with high probability.*

Proof. Due to Proposition 2, it suffices to analyze the distribution of path lengths in the Random Walk Model. It thus suffices to show that for any d sufficiently large, there exists a m depending only on d such that

$$\forall K \geq 1 : \qquad P(S_K > m \log K) \leq \frac{1}{K^d}, \tag{5.13}$$

for S_K as defined in (5.12). From Lemma 6, we know that the random variables $X_1, \ldots, X_{K-1}$ in S_K are mutually independent. Then using the Chernoff bound [126], we know that for any $m > 1$,

$$\begin{aligned} P(S_K > m(1 + H_{K-1})) &\leq \left(\frac{e^{m-1}}{m^m}\right)^{1+H_{K-1}} \\ &\leq \left(\frac{e^{m-1}}{m^m}\right)^{1+\log K} \\ &= (eK)^{m-1-m\log m} \end{aligned} \tag{5.14}$$

(5.14) is true since

$$\log K \leq H_{K-1} < \log K + 1$$

for all $K \geq 1$. We require this bound to be at most $1/K^d$, or

$$(eK)^{m-1-m\log m} \leq K^{-d}.$$

The above inequality is satisfied by $m \geq d$ for $d \geq e$. The Chernoff bound applies for all $K \geq 0$. So for any $d \geq e$, we can choose $m = d$ to satisfy (5.13). □

Corollary 1. *Assuming IF1 is mistake-free, then it plays $\mathcal{O}(K \log K)$ matches with high probability.*

Proof. The result immediately follows from Lemma 7 by noting that IF1 plays at most $\mathcal{O}(K)$ matches in each round. □

5.3.4 Regret Bound for Interleaved Filter 2

We first state our main regret bound for Interleaved Filter 2.

Theorem 4. *Running Algorithm 4 with $\mathcal{B} = \{b_1, \ldots, b_K\}$, time horizon T ($T \geq K$), and IF2 incurs expected generalized regret (and thus also weak and strong regret) bounded by*

$$\mathbf{E}[R_T] \leq \mathcal{O}\left(\mathbf{E}\left[R_T^{IF2}\right]\right) = \mathcal{O}\left(\frac{K}{\epsilon_{1,2}} \log T\right).$$

The proof follows immediately from combining Lemma 4, (5.6), and Lemmas 9 and 10 to follow. IF2 builds upon IF1 by additionally removing all empirically inferior bandits whenever the incumbant is defeated, which we call pruning. We begin by analyzing the pruning procedure. The following lemma could be informally summarized by saying that when IF2 produces a new incumbent b' and then eliminates a bandit b in the subsequent pruning step, we can conclude that b' is superior to b with $1 - (\delta T)$ confidence.

Algorithm 6 Interleaved Filter 2 (IF2)

Input: T, $\mathcal{B} = \{b_1, \ldots, b_K\}$
$\delta \leftarrow 1/(TK^2)$
Choose $\hat{b} \in \mathcal{B}$ randomly
$W \leftarrow \{b_1, \ldots, b_K\} \setminus \{\hat{b}\}$
$\forall b \in W$, maintain estimate $\hat{P}_{\hat{b},b}$ of $P(\hat{b} > b)$ according to (5.7)
$\forall b \in W$, maintain $1-\delta$ confidence interval $\hat{C}_{\hat{b},b}$ of $\hat{P}_{\hat{b},b}$ according to (5.8), (5.9)
while $W \neq \emptyset$ **do**
  for $b \in W$ **do**
    compare $\hat{b}$ and b
    update $\hat{P}_{\hat{b},b}$, $\hat{C}_{\hat{b},b}$
  end for
  while $\exists b \in W$ s.t. $\left(\hat{P}_{\hat{b},b} > 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b}\right)$ **do**
    $W \leftarrow W \setminus \{b\}$ *// $\hat{b}$ declared winner against b*
  end while
  if $\exists b' \in W$ s.t. $\left(\hat{P}_{\hat{b},b'} < 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b'}\right)$ **then**
    while $\exists b \in W$ s.t. $\hat{P}_{\hat{b},b} > 1/2$ **do**
      $W \leftarrow W \setminus \{b\}$ *//pruning*
    end while
    $\hat{b} \leftarrow b'$, $W \leftarrow W \setminus \{b'\}$ *// b' declared winner against $\hat{b}$ (new round)*
    $\forall b \in W$, reset $\hat{P}_{\hat{b},b}$ and $\hat{C}_{\hat{b},b}$
  end if
end while
$\hat{T} \leftarrow$ Total Comparisons Made
return $(\hat{b}, \hat{T})$

Lemma 8. *For all triples of bandits $b, b', \hat{b}$ such that $b \succ b'$, the probability that IF2 eliminates b in a pruning step in which b' wins a match against the incumbent bandit $\hat{b}$ (i.e. $\hat{P}_{\hat{b},b'} < 1/2$) while b is found to be empirically inferior to $\hat{b}$ (i.e. $\hat{P}_{\hat{b},b} > 1/2$) is at most δ.*

Proof. Let $X_1, X_2, \ldots$ denote an infinite sequence of i.i.d. Bernoulli random variables with $\mathbf{E}[X_i] = P(\hat{b} \succ b')$, and let $Y_1, Y_2, \ldots$ denote an infinite sequence of i.i.d. Bernoulli random variables with $\mathbf{E}[Y_i] = P(\hat{b} \succ b)$. We couple the outcomes of the comparisons performed by the algorithm to the sequences (X_i), (Y_i) in the obvious way: X_i (resp. Y_i) represents the outcome of the i^{th} comparison between $\hat{b}$ and

b' (resp. $\hat{b}$ and b) if the algorithm performs at least i comparisons of that pair of bandits; otherwise X_i (resp. Y_i) does not correspond to any comparison observed by the algorithm.

If b is eliminated by IF2 in a pruning step at the end of a match consisting of n comparisons between b' and the incumbent $\hat{b}$, then $X_1, \ldots, X_n$ represent the outcomes of the n matches between $\hat{b}$ and b' in that round, and $Y_1, \ldots, Y_n$ represent the outcomes of the n matches between $\hat{b}$ and b in that round. From the definition of confidence intervals in IF2 we know that $X_1 + \cdots + X_n < n/2 - \sqrt{4n \log(1/\delta)}$, whereas the definition of the pruning step implies that $Y_1 + \cdots + Y_n > n/2$. Thus, if we define $Z_i = Y_i - X_i$ for $i = 1, 2, \ldots$, then we have

$$Z_1 + \cdots + Z_n > \sqrt{4n \log(1/\delta)}. \tag{5.15}$$

To complete the proof of the lemma, we will show the probability that there exists an n satisfying (5.15) is at most δT.

The random variables $(Z_i)_{i=1}^{\infty}$ are i.i.d. and satisfy $|Z_i| \leq 1$. Furthermore, our assumption that $b \succ b'$ together with strong stochastic transitivity implies that

$$\mathbf{E}[Z_i] = P(\hat{b} \succ b) - P(\hat{b} \succ b') \leq 0.$$

By Hoeffding's inequality, for every n the probability that $\sum_{i=1}^{n} Z_i$ exceeds $\sqrt{4n \log(1/\delta)}$ is at most $\exp(-8n \log(1/\delta)/(4n)) = \delta^2$. Taking the union bound over $n = 1, 2, \ldots, T$, we find that the probability that there exists an n satisfying (5.15) is at most $\delta^2 T \leq \delta$, as claimed. □

Lemma 9. *The probability that IF2 makes a mistake resulting in the elimination of bandit b_1 is at most $1/T$.*

Proof. By Lemma 2, for every i the probability that b_1 is eliminated in a match against b_i is at most δ. A union bound over all i implies that the probability of

b_1 being eliminated by directly losing a match to some other bandit is at most $\delta(K-1)$. On the other hand, by Lemma 8, for all i, j the probability that b_1 is eliminated in a pruning step resulting from a match in which b_i defeats b_j is at most δ. A union bound over all i, j implies that the probability of b_1 being eliminated in a pruning step is at most $\delta(K-1)^2$. Summing these two bounds, the probability that IF2 makes a mistake resulting in the elimination of b_1 is at most $\delta[(K-1)+(K-1)^2] < \delta K^2 = 1/T$. □

For the remainder of this section, we analyze the behavior of IF2 when it is mistake-free. We will show that, in expectation, IF2 plays $O(K)$ matches and thus incurs expected regret bounded by

$$\mathcal{O}\left(\frac{K}{\epsilon_{1,2}}\log T\right).$$

Lemma 10. *Assuming IF2 is mistake free, then it plays $\mathcal{O}(K)$ matches in expectation.*

Proof. Let B_j denote a random variable counting the number of matches played by b_j when it is *not* the incumbant (to avoid double-counting). We can write B_j as

$$B_j = A_j + G_j,$$

where A_j indicates the number of matches played by b_j against b_i for $i > j$ (when the incumbant was inferior to b_j), and G_j indicates the number of matches played by b_j against b_i for $i < j$ (when the incumbant was superior to b_j). We can thus bound the expected number of matches played via

$$\sum_{j=1}^{K-1} \mathbf{E}[B_j] = \sum_{j=1}^{K-1} \mathbf{E}[A_j] + \mathbf{E}[G_j]. \tag{5.16}$$

By Lemma 6 and leveraging the Random Walk Model defined in Section 5.3.3, we can write $\mathbf{E}[A_j]$ as

$$\mathbf{E}[A_j] \leq 1 + \sum_{i=j+1}^{K-1} \frac{1}{i} = 1 + H_{K-1} - H_i,$$

where H_i is the harmonic sum.

We now analyze $\mathbf{E}[G_j]$. We assume the worst case that b_j does not lose a match (with $1-\delta$ confidence) to any superior incumbant b_i before the match concludes (b_i is defeated) unless $b_i = b_1$. We can thus bound $\mathbf{E}[G_j]$ using the probability that b_j is pruned at the conclusion of each round. Let $\mathcal{E}_{j,t}$ denote the event that b_j is pruned after the tth round in which the incumbant bandit is superior to b_j, conditioned on not being pruned in the first $t-1$ such rounds. Define $G_{j,t}$ to indicate the number of matches beyond the first $t-1$ played by b_j against a superior incumbant, conditioned on playing at least $t-1$ such matches. We can write $\mathbf{E}[G_{j,t}]$ as

$$\mathbf{E}[G_{j,t}] = 1 + P(\mathcal{E}_{j,t}^c)\mathbf{E}[G_{j,t+1}],$$

and thus

$$\mathbf{E}[G_j] \leq \mathbf{E}[G_{j,1}] \leq 1 + P(\mathcal{E}_{j,1}^c)\mathbf{E}[G_{j,2}]. \tag{5.17}$$

We know that $P(\mathcal{E}_{j,t}^c) \leq 1/2$ for all $j \neq 1$ and t. From Lemma 7, we know that $\mathbf{E}[G_{j,t}] \leq \mathcal{O}(K \log K)$ and is thus finite. Hence, we can bound (5.17) by the infinite geometric series $1 + 1/2 + 1/4 + \ldots = 2$.

We can thus write (5.16) as

$$\begin{aligned}\sum_{j=1}^{K-1} \mathbf{E}[A_j] + \mathbf{E}[G_j] &\leq \sum_{j=1}^{K-1} (1 + H_{K-1} - H_j) + 2(K-1) \\ &= \sum_{j=1}^{K-1} \left(1 + \sum_{i=j+1}^{K-1} \frac{1}{i}\right) + 2(K-1) \\ &= \sum_{j=1}^{K-1} (j-1)\frac{1}{j} + 3(K-1) = \mathcal{O}(K).\end{aligned}$$

□

5.3.5 Lower Bounds

We now show that the bound in Theorem 4 is information theoretically optimal up to constant factors. The proof is similar to the lower bound proof for the standard stochastic multi-armed bandit problem. However, since we make a number of assumptions not present in the standard case (such as a total ordering of $\mathcal{B}$), we present a simple self-contained lower bound argument, rather than a reduction from the standard case.

Theorem 5. *Any algorithm ϕ for the dueling bandits problem satisfies*

$$R_T^{\phi} = \Omega\left(\frac{K}{\epsilon}\log T\right),$$

where $\epsilon = \min_{b \neq b^*} P(b^* > b)$.

Here is a heuristic explanation of why we might suspect the theorem to be true. Rather than consider the general problem of identifying the best of K bandits, suppose we are given a bandit b, and asked to determine with probability at least $1 - 1/T$ whether b is the best bandit. (Intuitively, the regret incurred by the

optimal algorithm for this decision problem should be a lower bound on the regret incurred by the optimal algorithm for the general problem). We have seen that, given two bandits b_i and b_j with $P(b_i > b_j) = 1/2 + \epsilon$, we can identify the better bandit with probability at least $1 - 1/T$ after $O(\log T/\epsilon^2)$ comparisons. If this is in fact the minimum number of comparisons required, then we would suspect that any algorithm for the above decision problem that is uniformly good over all problem instances must perform $\Omega(\log T/\epsilon^2)$ comparisons involving each inferior bandit. We will see in Lemma 11 that this is in fact the case, and we begin by constructing the appropriate problem instance.

Fix $\epsilon > 0$ and define the following family of problem instances. In instance j, let b_j be the best bandit, and order the remaining bandits by their indices. That is, in instance j, we have $b_j \succ b_k$ for all $k \neq j$, and for $i, k \neq j$, we have $b_i \succ b_k$ whenever $i < k$. Given this ordering, define the winning probabilities by $P(b_i > b_k) = 1/2 + \epsilon$ whenever $b_i \succ b_k$. Note that this construction yields a valid problem instance, i.e. one that satisfies (5.4), (5.5).

Let q_j be the distribution on T-step histories induced by a given algorithm ϕ under instance j, and let $n_{j,T}$ be the number of comparisons involving bandit b_j scheduled by ϕ up to time T. Using these instances, we prove Lemma 11, from which Theorem 5 follows.

Lemma 11. *Let ϕ be an algorithm for the dueling bandits problem such that*

$$R_T^{\phi} = o(T^a) \tag{5.18}$$

for all $a > 0$. Then for all j,

$$\mathbf{E}_{q_1}[n_{j,T}] = \Omega\left(\frac{\log T}{\epsilon^2}\right).$$

Lemma 11 formalizes the intuition given above, in that any algorithm whose regret is $o(T^a)$ over all problem instances must make $\Omega(\log T/\epsilon^2)$ comparisons involving each inferior bandit, in expectation. The proof is motivated by Lemma 5 of [99].

Proof. Fix an algorithm ϕ satisfying assumption (5.18), and fix $0 < a < 1/2$. Define the event $\mathcal{E}_j = \{n_{j,T} < \log(T)/\epsilon^2\}$, and let $J = \{j : q_1(\mathcal{E}_j) < 1/3\}$. For each $j \in J$, we have by Markov's inequality that

$$\mathbf{E}_{q_1}[n_{j,T}] \geq q_1(\mathcal{E}_j^c)(\log(T)/\epsilon^2) = \Omega\left(\frac{\log T}{\epsilon^2}\right),$$

so it remains to show that $\mathbf{E}_{q_1}[n_{j,T}] = \Omega(\log T/\epsilon^2)$ for each $j \notin J$. For any j, we know that under q_j, the algorithm ϕ incurs regret ϵ for every comparison involving a bandit $b \neq b_j$. This fact together with the assumption (5.18) on ϕ implies that $\mathbf{E}_{q_j}[T - n_{j,T}] = o(T^a)$. Using this fact and Markov's inequality, we have

$$\begin{aligned} q_j(\mathcal{E}_j) &= q_j(\{T - n_{j,T} > T - \log(T)/\epsilon^2\}) \\ &\leq \frac{\mathbf{E}_{q_j}[T - n_{j,T}]}{T - \log(T)/\epsilon^2} = o(T^{a-1}), \end{aligned}$$

and so choosing T sufficiently large shows that $q_j(\mathcal{E}_j) < 1/3$ for each j (and in particular, that $1 \in J$ by construction). Now by Lemma 6.3 of [91], we have that for any event $\mathcal{E}$ and distributions p, q with $p(\mathcal{E}) \geq 1/3$ and $q(\mathcal{E}) < 1/3$,

$$KL(p; q) \geq \frac{1}{3} \ln\left(\frac{1}{3q(\mathcal{E})}\right) - \frac{1}{e}.$$

For each $j \notin J$, we may apply this lemma with q_1, q_j, and the event $\mathcal{E}_j$, to show

$$\begin{aligned} KL(q_1; q_j) &\geq \frac{1}{3} \ln\left(\frac{1}{3o(T^{a-1})}\right) - \frac{1}{e} \\ &= \Omega(\log T). \end{aligned} \tag{5.19}$$

On the other hand, by the chain rule for KL divergence [47], we have

$$\begin{aligned} KL(q_1; q_j) &\leq \mathbf{E}_{q_1}[n_{j,T}]\, KL(1/2+\epsilon; 1/2-\epsilon) \\ &\leq 16\epsilon^2 \mathbf{E}_{q_1}[n_{j,T}], \end{aligned} \tag{5.20}$$

where we use the shorthand $KL(1/2+\epsilon; 1/2-\epsilon)$ to denote the KL-divergence between two Bernoulli distributions with parameters $1/2+\epsilon$ and $1/2-\epsilon$, respectively. The first inequality follows from the fact that the distribution on the outcome of a comparison will differ under distributions q_1 and q_j only if the comparison involves bandit b_j, and the second inequality follows from a standard result on the KL divergence between two Bernoulli distributions. Combining (5.19) and (5.20) shows that $\mathbf{E}_{q_1}[n_{j,T}] = \Omega(\log T/\epsilon^2)$ for each $j \notin J$, which proves the lemma. □

Proof of Theorem 5. Let ϕ be any algorithm for the dueling bandits problem. If ϕ does not satisfy the hypothesis of Lemma 11, the theorem holds trivially. Otherwise, on the problem instance specified by q_1, ϕ incurs regret at least ϵ every time it plays a match involving $b_j \neq b_1$. It follows from Lemma 11 that

$$R_T^\phi \geq \sum_{j \neq 1} \epsilon \mathbf{E}_{q_1}[n_{j,T}] = \Omega\left(\frac{K}{\epsilon} \log T\right).$$

□

5.4 Related Work

Regret-minimizing algorithms for multi-armed bandit problems and their generalizations have been intensively studied for many years, both in the stochastic [108] and non-stochastic [13] cases. The vast literature on this topic includes algorithms whose regret is within a constant factor of the information-theoretic lower

bound in both the stochastic case [12] and the non-stochastic case [10]. Our use of upper confidence bounds in designing algorithms for the dueling bandits problem is prefigured by their use in the multi-armed bandit algorithms that appear in [11, 12, 108].

Upper confidence bounds are also central to the design of multi-armed bandit problems in the PAC setting [60, 122], where the algorithm's objective is to identify an arm that is ε-optimal with probability at least $1 - \delta$. Our work adopts a very different feedback model (pairwise comparisons rather than direct observation of payoffs) and a different objective (regret minimization rather than the PAC objective) but there are clear similarities between our proposed algorithms and the Successive Elimination and Median Eliminiation algorithms developed for the PAC setting in [60]. There are also some clear differences between the algorithms: in our setting, the highly suboptimal arms must be eliminated quickly (before sampling more that ϵ^{-2} times). In the Successive/Median Elimination algorithms, every arm is sampled at least ϵ^{-2} times. The need to eliminate highly suboptimal arms quickly is specific to the regret minimization setting and exerts a strong influence on the design of the algorithm; in particular, it motivates the interleaved structure as explained above.

The difficulty of the dueling bandits problem stems from the fact that the algorithm has no way of directly observing the costs of the actions it chooses. It is an example of a *partial monitoring problem*, a class of regret-minimization problems defined in [34], in which an algorithm (the "forecaster") chooses actions and then observes feedback signals that depend on the actions chosen by the forecaster and by an unseen opponent (the "environment"). This pair of actions also determines a loss, which is not revealed to the forecaster but is used in defining the forecaster's

regret. Under the crucial assumption that the feedback matrix has high enough rank that its row space spans the row space of the loss matrix (which is required in order to allow for a Hannan consistent forecaster) the results of [34] show that there is a forecaster whose regret is bounded by $O(T^{2/3})$ against a non-stochastic (adversarial) environment, and that there exist partial monitoring problems for which this bound cannot be improved. Our dueling bandits problem is a special case of the partial monitoring problem. In particular, our environment is stochastic rather than adversarial, and thus our regret bound exhibits much better (i.e., logarithmic) dependence on T.

Banditized online learning problems based on absolute rewards (of individual actions) have been previously studied in the context of web advertising [129, 111]. In that setting, clear explicit feedback is available in the form of (expected) revenue. We study settings where such absolute measures are unavailable or unreliable.

Our work is also closely related to the literature on computing with noisy comparison operations [1, 15, 61, 91], in particular the design of tournaments to identify the maximum element in an ordered set, given access to noisy comparators. All of these papers assume unit cost per comparison, whereas we charge a different cost for each comparison depending on the pair of elements being compared. In the unit-cost-per-comparison model, and assuming that every comparison has ϵ probability of error regardless of the pair of elements being compared, Feige et al. [61] presented sequential and parallel algorithms that achieve the information-theoretically optimal expected cost (up to constant factors) for many basic problems such as sorting, searching, and selecting the maximum. The upper bound for noisy binary search has been improved in a recent paper [15] that achieves the information-theoretic optimum up to a $1 + o(1)$ factor. When the probabil-

ity of error depends on the pair of elements being compared (as in our dueling bandits problem), Adler et al. [1] and Karp and Kleinberg [91] present algorithms that achieve the information-theoretic optimum (up to constant factors) for the problem of selecting the maximum and for binary search, respectively. Our results can be seen as extending this line of work to the setting of regret minimization. It is worth noting that the most efficient algorithms for selecting the maximum in the model of noisy comparisons with unit cost per comparison [1, 61] are not suitable in the regret minimization setting considered here, because they devote undue effort to comparing elements that are far from the maximum.

Learning based on pairwise comparisons is well studied in the (off-line) supervised learning setting called learning to rank (also see Section 2.2.3). Typically, a preference function is first learned using a set of i.i.d. training examples, and subsequent predictions are made to minimize the number of mis-ranked pairs (e.g., [44]). Most prior work assume access to a training set with absolute labels (e.g., of relevance or utility) on individual examples, with pairwise preferences generated using pairs of inputs with labels from different ordinal classes (e.g., [5, 14, 66, 75, 85, 119]). In the case where there are exactly two label classes, this becomes the so-called bipartite ranking problem [5, 14], which is a more general version of learning to optimize ROC-Area [75, 85, 119].

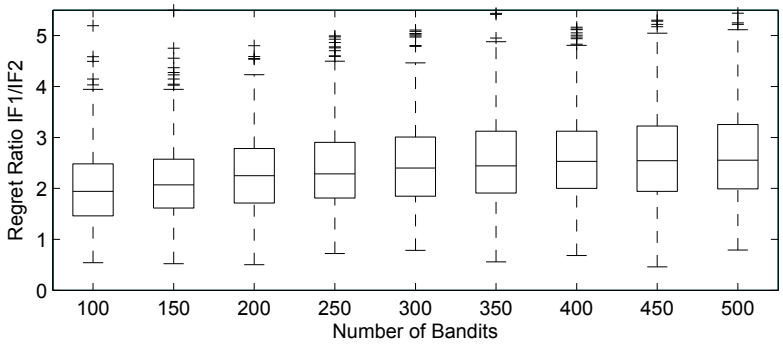

Figure 5.1: Comparing regret ratio between IF1 and IF2 in worst-case simulations.

5.5 Experiments

5.5.1 Synthetic Simulations

We performed numerical simulations on two synthetic problem instances. The first set of simulations used the worst-case instance from the lower bound proof of Theorem 5. In this instance, $P(b_i > b_j) = 1/2 + \epsilon$ whenever $i < j$. For this experiment, we fixed $\epsilon = 0.1$ and the time horizon $T = 10^7$. We varied K from 100 to 500 in increments of 50, and for each value of K, we performed 500 simulations of both IF1 and IF2. In Figure 5.1, we plot the ratio of the regret incurred by IF1 and IF2 (which we henceforth also call the regret ratio).

For the second set of simulations, we generated random problem instances according to a Bradley-Terry model with uniformly random weights. For normalization purposes, we then modified each problem instance to ensure that the best bandit had a winning probability of at least $1/2 + \epsilon$ against all other bandits. The details of the procedure are as follows. For each value of K, we generated $K - 1$ random numbers $w_2, \ldots, w_K$ sampled independently from the uniform distribution on $(0, 1)$. To define w_1, we found the largest weight $w_{\max} = \max\{w_2, \ldots, w_K\}$, and

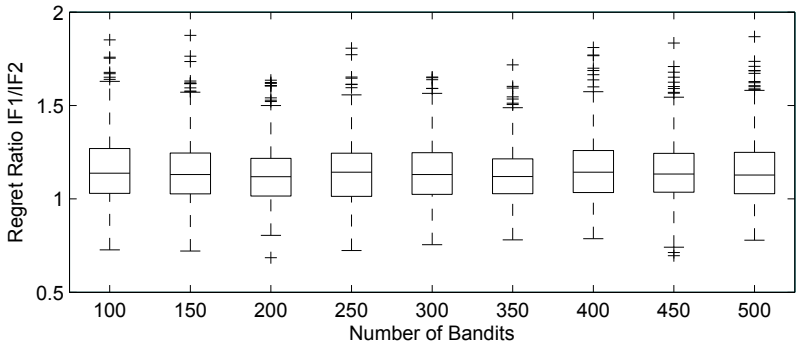

Figure 5.2: Comparing regret ratio between IF1 and IF2 in random case simulations.

defined $w_1 = w_{\max}(1+2\epsilon)/(1-2\epsilon)$. We then defined $P(b_i > b_j) = w_i/(w_i + w_j)$, so that for all $i \neq 1$,

$$P(b_1 > b_i) = \frac{w_1}{w_1 + w_i} \geq \frac{w_1}{w_1 + w_{\max}} = \frac{1}{2} + \epsilon.$$

Note that this is the Bradley-Terry model discussed in Section 5.2, which satisfies the modeling assumptions introduced in that section. We fixed $\epsilon = 0.1$ and $T = 10^7$, and performed 500 simulations of IF1 and IF2 on each of the randomly generated instances. We plot the regret ratio of IF1 versus IF2 in Figure 5.2.

For the worst-case simulations, we see that IF2 outperforms IF1, and that the median of the regret ratio increases logarithmically with K. For the random-case simulations, we also see that IF2 outperforms IF1, but the regret ratio does not increase with K as in the worst-case simulations. Intuitively, IF1 and IF2 incur a large amount of regret during matches in which $P(b_i > b_j)$ is close to 1/2. In the worst-case problem instance, this is guaranteed to be true for every match by construction. Consequently, each pruning step performed by IF2 reduces the total regret incurred by a significant amount by eliminating a high-cost match that would otherwise be played. In contrast to the worst-case instances, we expect that in the random-case, many matches will have $P(b_i > b_j)$ far from 1/2, and thus will

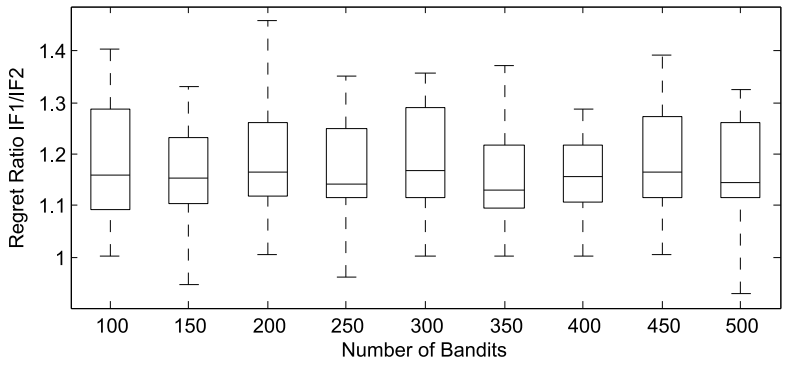

Figure 5.3: Comparing regret ratio between IF1 and IF2 in web search simulations.

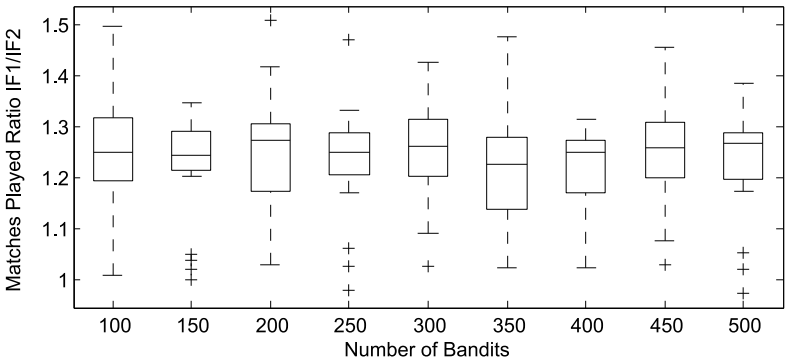

Figure 5.4: Comparing matches played ratio between IF1 and IF2 in web search simulations.

contribute little to the total regret. A pruning step that eliminates such a match will have little effect on the total regret, and so we should expect the regret of IF1 and IF2 to be more similar in the random-case than in the worst-case, which fits our empirical results.

5.5.2 Web Search Simulations

For a more realistic simulation, we leveraged a real Web Search dataset (courtesy of Chris Burges at Microsoft Research). The idea is to simulate users issuing queries by sampling from queries in the dataset. For each query, the two competing

retrieval functions will produce rankings, after which the "user" will randomly prefer one ranking over the other. User preferences are modeled probabilistically using the logistic transfer function and NDCG@10, which is a measure used for evaluating the quality of rankings in information retrieval tasks (see [58]).

The compatibility between a document/query pair is represented using 367 features. A standard retrieval function computes a score for each document based on these features, with the final ranking resulting from sorting by the scores. We can then use that ranking to compute NDCG@10. For simplicity, we considered only linear functions w, so that the score for document x is $w^T x$. Any particular bandit corresponds to a particular weight vector.

We varied the number of bandits (retrieval functions) K from 100 to 500 in increments of 50. For each experimental setting, we randomly selected K retrieval functions from a pool of 1000 retrieval functions. For each value of K, we used 25 experimental settings with 25 trials per setting. We fixed $T = 10^7$ for all settings, since our primary goal in this experiment is to compare the performance of IF1 and IF2. We used strong regret (5.1) to measure performance.

Figure 5.3 shows a box plot of the regret ratio for IF1 and IF2. Since different collections of retrieval functions yield different performances (due to differences in the distinguishability between the bandits), it is more informative to compare the ratio of regret on the same initial conditions, which we again call the regret ratio.[9] We can see that IF2 consistently outperforms IF1, however the performance ratio does not scale as $\log(K)$ as implied by our worst case bounds.

Intuitively, there are two conditions that must be satisfied for IF2 to improve

[9]Both IF1 and IF2 start with the same initial incumbant bandit and the same (randomly selected) permutation ordering over the remaining bandits to use for when interleaving matches in round robin fashion.

by a logarithmic factor over IF1. First, a logarithmic number of rounds must be played (i.e., we must consider a logarithmic number of candidate bandits). Second, within each round, most of the bandits must not be confidently eliminated from consideration so that they can be eliminated via the pruning procedure in IF2. Satisfying both of these conditions would imply IF1 playing a logarithmic factor more matches than IF2. In the web search dataset, we observe neither condition being strongly satisfied. In all settings, only a small number of rounds are played (typically between 2 and 4) for all values of K (which admittedly only ranges up to 500 in our experiments). Futhermore, in many rounds, a substantial fraction of the bandits are confidently eliminated from consideration before the conclusion of the round. This is summarized in Figure 5.4, which shows a box plot of the ratio of matches played between IF1 and IF2. Nonetheless, we can see that IF2 can offer real practical improvements over IF1, although the difference in performance is perhaps not as dramatic as suggested by the worst case analysis.

5.6 Discussion

The Dueling Bandits Problem is appealing due to not only its simplicity, but also its practical applicability. In the discrete setting, our Interleaved Filter algorithms are guaranteed to provide good performance so long as there exists a comparison oracle which satisfies the assumptions described in Section 5.2.

But while the methods presented in this chapter demonstrate significant progress towards practical interactive algorithms for real information systems, significant challenges remain. Most obviously, do such comparison oracles which satisfy our modeling assumptions actually exist in practice? And if not, then can we design

methods that are robust to using oracles which violate these assumptions in some quantifiable way?

From a theoretical standpoint, questions also remain. For instance, while Interleaved Filter 2 achieves expected regret that is information-theoretically optimal (up to constant factors), it is an open question whether there exists algorithms that can achieve this performance with high probability. The formulation of the Dueling Bandits Problem can also be naturally extended along many other standard directions, including incorporating shifting user interests, unknown time horizons, and adversarial behavior.

From a practical standpoint, two additional related modeling questions arise. The first is how to deal with context. Different retrieval functions may perform better for different queries or usage contexts, and we may want our algorithm to be able to choose which retrieval functions to compare depending on that context. The current formulation of the Dueling Bandits Problem ignores contextual information. This issue has been explored in the standard multi-armed bandit setting in two ways. The first is to assume a collection of "experts" that gives advice on which bandit (or retrieval function, in our case) is best for a given context [12]. The second, and more general, setting is to assume a (continuous) hypothesis class of classifiers that predicts which bandit is best for a given context [111]. Both problem settings can be easily reformulated into a dueling bandits setting.

The second modeling question is how to deal with large strategy spaces (i.e., large K). Since this subsumes the aforementioned problem of dealing with contextual information (by treating the strategy space as the Cartesian product of the original strategy space and the space of contexts), we focus here on a complementary sub-problem. Note that the regret bound for Interleaved Filter 2 is linear in K,

which can be prohibitively expensive for even moderately large values of K (e.g., one thousand or one million). The most common approach to dealing with such issues in the standard multi-armed bandit setting is to assume additional structure in the strategy space, such as a metric [100] or a hierarchical decomposition [129]. This is a largely unexplored problem area in the dueling bandits setting., and the specific modeling considerations will depend upon the application.

CHAPTER 6

THE DUELING BANDITS PROBLEM FOR CONTINUOUS PARAMETER SPACES

In this chapter, we investigate an instance of the Dueling Bandits Problem that deals with a continuous space of bandits $\mathcal{W}$ [185]. This setting is very practical as many information systems employ continuously parameterized retrieval functions, and interactively optimizing those parameters can be naturally modeled using the *continuum-armed* Dueling Bandits Problem described in the following.

More specifically, we consider the setting where $\mathcal{W}$ contains the origin and is compact, convex, and contained in a d-dimensional ball of radius R. Like in Chapter 5, we assume that any single comparison between two points w and w' (e.g., individual retrieval functions) is determined independently of all other comparisons with probability

$$P(w \succ w') = \frac{1}{2} + \epsilon(w, w'), \tag{6.1}$$

where $\epsilon(w, w') \in [-1/2, 1/2]$. In the search example, $P(w \succ w')$ refers to the fraction of users who prefer the results produced by w over those of w'. One can regard $\epsilon(w, w')$ as the distinguishability between w and w'. Algorithms learn only via observing comparison results (e.g., from interleaving [140]).

We quantify the performance of an on-line algorithm using the strong regret formulation from Chapter 5,[1]

$$R_T = \sum_{t=1}^{T} \epsilon(w^*, w_t) + \epsilon(w^*, w'_t), \tag{6.2}$$

[1]Our results also apply, with little modification, to other regret formulations proposed in Chapter 5.

where w_t and w'_t are the two points selected at time t, and w^* is the best point known only in hindsight. Note that the algorithm is allowed to select two identical points, so selecting $w_t = w'_t = w^*$ accumulates no additional regret. In the search example, regret corresponds to the fraction of users who would prefer the best retrieval function w^* over the selected ones w_t and w'_t.

6.1 Modeling Assumptions

We further assume the existence of a differentiable, strictly concave value function $v : \mathcal{W} \to \mathcal{R}$. This function reflects the intrinsic quality of each bandit/point in $\mathcal{W}$, and is never directly observed. Since v is strictly concave, there exists a unique maximum $v(w^*)$. Probabilistic comparisons are made using a link function $\sigma : \mathcal{R} \to [0, 1]$, and are defined as

$$P(w \succ w') = \sigma(v(w) - v(w')).$$

Thus $\epsilon(w, w') = \sigma(v(w) - v(w')) - 1/2$.

Link functions behave like cumulative distribution functions (monotonic increasing, $\sigma(-\infty) = 0$, and $\sigma(\infty) = 1$). We consider only link functions which are rotation-symmetric ($\sigma(x) = 1 - \sigma(-x)$) and have a single inflection point at $\sigma(0) = 1/2$. This implies that $\sigma(x)$ is convex for $x \leq 0$ and concave for $x \geq 0$. One common link function is the logistic function $\sigma_L(x) = 1/(1 + \exp(-x))$.

We finally make two smoothness assumptions. First, σ is L_σ-Lipschitz, and v is L_v-Lipschitz. That is, $|\sigma(a) - \sigma(b)| \leq L_\sigma \|a - b\|$. Thus $\epsilon(\cdot, \cdot)$ is L-Lipschitz in both arguments, where $L = L_\sigma L_v$. We further assume that L_σ and L_v are the least possible. Second, σ is second order L_2-Lipschitz, that is, $|\sigma'(a) - \sigma'(b)| \leq L_2 \|a - b\|$. These relatively mild assumptions provide sufficient structure for showing sublinear

Algorithm 7 Dueling Bandit Gradient Descent

Input: γ, δ, w_1
for query q_t ($t = 1..T$) **do**
  Sample unit vector u_t uniformly.
  $w'_t \leftarrow \mathbf{P}_{\mathcal{W}}(w_t + \delta u_t)$ *//projected back into* $\mathcal{W}$
  Compare w_t and w'_t
  if w'_t wins **then**
    $w_{t+1} \leftarrow \mathbf{P}_{\mathcal{W}}(w_t + \gamma u_t)$ *//also projected*
  else
    $w_{t+1} \leftarrow w_t$
  end if
end for

regret.

Note that these are stronger assumptions than the ones made in Chapter 5, since they immediately imply strong stochastic transitivity (5.4) and stochastic triangle inequality (5.5).

6.2 Dueling Bandit Gradient Descent

We now present an algorithm and analysis which build upon methods for online convex optimization [192, 98, 65]. This method is compatible with many existing classes of retrieval functions, and we provide theoretical regret bounds and an experimental evaluation.

Our algorithm, Dueling Bandit Gradient Descent (DBGD), is described in Algorithm 7. DBGD maintains a candidate w_t and compares it with a neighboring point w'_t along a random direction u_t. If w'_t wins the comparison, then an update is taken along u_t, and then projected back into $\mathcal{W}$ (denoted by $\mathbf{P}_{\mathcal{W}}$).

DBGD requires two parameters which can be interpreted as the exploration

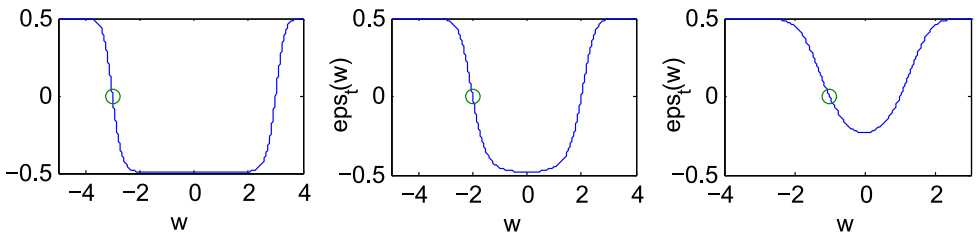

Figure 6.1: Example relative loss functions ($\epsilon_t(w) \equiv \epsilon(w_t, w)$) using the logistic link function, $\mathcal{W} \subseteq R$, and value function $v(w) = -w^2$, for $w_t = -3, -2, -1$. Note that the functions are convex in the area around $w^* = 0$.

(δ) and exploitation (γ) step sizes. The latter is required for all gradient descent algorithms. Since DBGD probes for descent directions randomly, this introduces a gradient estimation error that depends on δ (discussed Section 6.2.2). We will show in Theorem 7 that, for suitable δ and γ, DBGD achieves sublinear regret in T,

$$\mathbf{E}[R_T] \leq 2\lambda_T T^{3/4}\sqrt{26RdL},$$

where λ_T approaches 1 from above as T increases. For example, when $T > \frac{64R^2d^2L_v^4L_2^4}{13^2L^2L_\sigma^4}$, then $\lambda_T < 2$.

Making an additional convexity assumption[2] described in Theorem 9 yields a much simpler result,

$$\mathbf{E}[R_T] \leq 2T^{3/4}\sqrt{10RdL}.$$

To analyze DBGD, we first define relative loss as

$$\epsilon_t(w) \equiv \epsilon(w_t, w), \tag{6.3}$$

which is the distinguishability between w_t and any other point. We will also define $\epsilon^*(w)$ as

$$\epsilon^*(w) \equiv \epsilon(w^*, w). \tag{6.4}$$

[2]The assumption currently lacks theoretical justification, but is observed empirically in many settings.

This relative loss function is depicted pictorally in Figure 6.1 for the logistic link function and $v(w) = -w^2$.

Analysis Approach. Our analysis follows two conceptual phases. We first present basic results demonstrating the feasibility of performing gradient descent on the relative loss functions ϵ_t (6.3). These results include proving that ϵ_t is partially convex,[3] and how pairwise comparisons can yield good gradient estimates. We then build on existing results [192, 65] to show that DBGD minimizes our regret formulation (6.2). We begin by observing that ϵ_t is partially convex.

Observation 2. *For link functions $\sigma(x)$ and value functions $v(w)$ satisfying assumptions from Section 6.1, $\epsilon_t(w)$ is partially convex for $w_t \neq w^*$.*

Proof. Define $W_t = \{w : v(w) \geq v(w_t)\}$, which has a non-empty interior for $w_t \neq w^*$. For $a, b \in W_t$ and $\beta \in [0, 1]$ we know that

$$v(\beta a + (1-\beta)b) \geq \beta v(a) + (1-\beta)v(b),$$

since v is concave. We then write $\epsilon_t(\beta a + (1-\beta)b)$ as

$$\begin{aligned}
&= \sigma(v(w_t) - v(\beta a + (1-\beta)b)) - 1/2 \\
&\leq \sigma(v(w_t) - \beta v(a) - (1-\beta)v(b)) - 1/2 \\
&\leq \beta\sigma(v(w_t) - v(a)) + (1-\beta)\sigma(v(w_t) - v(b)) - 1/2 \\
&= \beta\epsilon_t(a) + (1-\beta)\epsilon_t(b)
\end{aligned}$$

The first inequality follows from monotonicity of $\sigma(x)$. The second inequality holds since $\sigma(x)$ is convex for $x \leq 0$ (holds for $a, b \in W_t$). Since W_t is convex (due to concavity of v), we conclude that ϵ_t is partially convex. □

[3] A function $f : \mathcal{W} \to R$ is partially convex if there is a convex region with a non-empty interior and containing w^* where f is convex.

6.2.1 Estimating Gradients

We now elaborate on the update procedure used by DBGD. Flaxman et al. [65] observed that

$$\nabla c_t(w_t) \approx \mathbf{E}_u[c_t(w_t + \delta u)u]\frac{d}{\delta}, \tag{6.5}$$

where $\delta > 0$, d denotes the dimensionality, and u is a uniformly random unit vector. Let $X_t(w)$ denote the event of w winning a comparison with w_t:

$$X_t(w) = \begin{cases} 1 & \text{w.p. } 1 - P(w_t \succ w) \\ 0 & \text{w.p. } P(w_t \succ w) \end{cases}. \tag{6.6}$$

We can model the update in DBGD (ignoring γ) as

$$X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u_t))u_t,$$

which we now show, in expectation, matches the RHS of (6.5) (ignoring d/δ) with an additional projection.

Lemma 12. *Let*

$$c_t(w) = P(w_t \succ w) = \epsilon_t(w) + 1/2.$$

Then for $\delta > 0$ and uniformly random unit vector u,

$$\mathbf{E}_{X_t,u}[X_t(\boldsymbol{P}_{\mathcal{W}}(w_t + \delta u))u] = -\mathbf{E}_u[c_t(\boldsymbol{P}_{\mathcal{W}}(w_t + \delta u))u].$$

Proof. Let $\mathbb{S}$ denote the unit sphere. Then we see that $\mathbf{E}_{X_t,u}[X_t(w_t + \delta u)u]$ can be

written as

$$= \mathbf{E}_u[\mathbf{E}_{X_t}[X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u))|u]u]$$

$$= \int_{\mathbb{S}} \mathbf{E}_{X_t}[X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u))|u]udu$$

$$= \int_{\mathbb{S}}(1 - c_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u)))udu$$

$$= 0 - \int_{\mathbb{S}} c_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u))udu$$

$$= -\mathbf{E}_u[c_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u))u]$$

□

6.2.2 Gradient Quality & Function Smoothing

We now characterize the quality of the proposed gradient approximation (6.5). Let $\hat{c}_t$ denote a smoothed version of some function c_t,

$$\hat{c}_t(w) = \mathbf{E}_{x \in \mathcal{B}}[c_t(\mathbf{P}_{\mathcal{W}}(w + \delta x))],$$

where x is selected uniformly within the unit ball $\mathcal{B}$. We can show using Stokes Theorem that our sampled gradient direction is an unbiased estimate of $\nabla \hat{c}_t$.

Lemma 13. *Fix $\delta > 0$, over random unit vectors u,*

$$\mathbf{E}_{u \in \mathbb{S}}[c_t(\boldsymbol{P}_{\mathcal{W}}(w + \delta u))u] = \frac{\delta}{d} \nabla \hat{c}_t(w),$$

where d is the dimensionality of w, and $\mathbb{S}$ denotes the unit sphere (of dimensionality d). (Analogous to Lemma 2.1 of [65])

Proof. (Adapted from [65].) For $d = 1$, the fundamental theorem of calculus implies

$$\frac{d}{dw} \int_{-\delta}^{\delta} c_t(\mathbf{P}_{\mathcal{W}}(w + x))dx = c_t(\mathbf{P}_{\mathcal{W}}(w + \delta)) - c_t(\mathbf{P}_{\mathcal{W}}(w - \delta)).$$

Using Stokes Theorem, we can write the d-dimensional generalization as

$$\nabla \int_{\delta\mathcal{B}} c_t(\mathbf{P}_{\mathcal{W}}(w+x))dx = \int_{\delta\mathbb{S}} c_t(\mathbf{P}_{\mathcal{W}}(w+u))\frac{u}{\|u\|}du. \tag{6.7}$$

By definition, we have

$$\hat{c}_t(w) = \mathbf{E}_{x\in\mathcal{B}}[c_t(\mathbf{P}_{\mathcal{W}}(w+\delta x))] = \frac{\int_{\delta\mathcal{B}} c_t(\mathbf{P}_{\mathcal{W}}(w+\delta x))dx}{\text{volume}(\delta\mathcal{B})}, \tag{6.8}$$

and

$$\mathbf{E}_{u\in\mathbb{S}}[c_t(\mathbf{P}_{\mathcal{W}}(w+\delta u)u] = \frac{\int_{\delta\mathbb{S}} c_t(\mathbf{P}_{\mathcal{W}}(w+u))\frac{u}{\|u\|}du}{\text{area}(\delta\mathbb{S})}. \tag{6.9}$$

Combining (6.7), (6.8), (6.9), and the fact that ratio of volume to surface area of a d-dimensional ball of radius δ is δ/d concludes the proof. □

Combining Lemma 12 and Lemma 13 implies that DBGD is essentially performing gradient descent over

$$\hat{\epsilon}_t(w) = \mathbf{E}_{x\in\mathcal{B}}[\epsilon_t(\mathbf{P}_{\mathcal{W}}(w+\delta x))]. \tag{6.10}$$

Note that $|\hat{\epsilon}_t(w) - \epsilon_t(w)| \leq \delta L$, and that $\hat{\epsilon}_t$ is parameterized by δ (suppressed for brevity). Hence, good regret bounds defined on $\hat{\epsilon}_t$ imply good bounds defined on ϵ_t, with δ controlling the difference.

One concern is that $\hat{\epsilon}_t$ might not be convex at w_t. Observation 2 showed that ϵ_t is convex at w_t, and thus satisfies $\epsilon_t(w_t) - \epsilon_t(w^*) \leq \nabla\epsilon_t(w_t)\cdot(w_t - w^*)$. We now show that $\hat{\epsilon}_t(w_t)$ is "almost convex" in a specific way.

Theorem 6. *For λ defined as*

$$\lambda = \frac{L_\sigma}{L_\sigma - \delta L_v L_2}, \tag{6.11}$$

and $\delta \in \left(0, \frac{L_\sigma}{L_v L_2}\right)$, then

$$\hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*) \leq \lambda\nabla\hat{\epsilon}_t(w_t)\cdot(w_t - w^*) + (3+\lambda)\delta L.$$

Proof. First define $w_{t,\delta x} \equiv \mathbf{P}_{\mathcal{W}}(w_t + \delta x)$, and also $\epsilon_{t,\delta x}(w) \equiv \epsilon(w_{t,\delta x}, w)$. We rewrite $\hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)$ as

$$= \mathbf{E}_{x\in\mathcal{B}}\left[\epsilon_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta x)) - \epsilon_t(\mathbf{P}_{\mathcal{W}}(w^* + \delta x))\right]$$

$$\leq \mathbf{E}_{x\in\mathcal{B}}\left[\epsilon_{t,\delta x}(w_{t,\delta x}) - \epsilon_{t,\delta x}(w^*)\right] + 3\delta L \tag{6.12}$$

$$\leq \mathbf{E}_{x\in\mathcal{B}}\left[\nabla\epsilon_{t,\delta x}(w_{t,\delta x}) \cdot (w_{t,\delta x} - w^*)\right] + 3\delta L \tag{6.13}$$

where (6.12) follows from ϵ being L-Lipschitz, and (6.13) follows from $w_{t,\delta x}$ and w^* both being in the convex region of $\epsilon_{t,\delta x}$. Now define $\sigma_t(y) \equiv \sigma(v(w_t) - y)$, and $\sigma_{t,\delta x}(y) \equiv \sigma(v(w_{t,\delta x}) - y)$. We can see that

$$\nabla\epsilon_t(w_{t,\delta x}) = \sigma_t'(v(w_{t,\delta x}))\nabla v(w_{t,\delta x}).$$

and similarly

$$\nabla\epsilon_{t,\delta x}(w_{t,\delta x}) = \sigma_{t,\delta x}'(v(w_{t,\delta x}))\nabla v(w_{t,\delta x}).$$

We can then write (6.13) as

$$= \mathbf{E}_x\left[\sigma_{t,\delta x}'(w_{t,\delta x})\nabla v(w_{t,\delta x}) \cdot (w_{t,\delta x} - w^*)\right] + 3\delta L. \tag{6.14}$$

We know that both $\sigma_{t,\delta x}'(y) \leq 0$ and $\sigma_t'(y) \leq 0$, and

$$\sigma_{t,\delta x}'(v(w_{t,\delta x})) = -L_\sigma,$$

since that is the inflection point. Thus

$$-L_\sigma \leq \sigma_t'(v(w_{t,\delta x})) \leq -L_\sigma + \delta L_v L_2,$$

which follows from σ being second order L_2-Lipschitz. Since $\epsilon_{t,\delta x}(w_{t,\delta x}) - \epsilon_{t,\delta x}(w^*) \geq 0$, the term inside the expectation in (6.14) is also non-negative. Using our defini-

tion of λ (6.11), we can write (6.14) as

$$\begin{aligned}
&\leq \mathbf{E}_x \left[\lambda \sigma_t'(w_{t,\delta x}) \nabla v(w_{t,\delta x}) \cdot (w_{t,\delta x} - w^*)\right] + 3\delta L \\
&= \mathbf{E}_x \left[\lambda \nabla \epsilon_t(w_{t,\delta x}) \cdot (w_{t,\delta x} - w^*)\right] + 3\delta L \\
&= \mathbf{E}_x \left[\lambda \nabla \epsilon_t(w_{t,\delta x}) \cdot (w_{t,\delta x} - w_t + w_t - w^*)\right] + 3\delta L \\
&\leq \mathbf{E}_x \left[\lambda \nabla \epsilon_t(w_{t,\delta x}) \cdot (w_t - w^*)\right] + (3+\lambda)\delta L \qquad (6.15) \\
&= \lambda \nabla \hat{\epsilon}_t(w_t) \cdot (w_t - w^*) + (3+\lambda)\delta L
\end{aligned}$$

where (6.15) follows from observing that

$$\mathbf{E}_x \left[\nabla \epsilon_t(w_{t,\delta x}) \cdot (w_{t,\delta x} - w_t)\right] \leq \mathbf{E}_x \left[\|\nabla \epsilon_t(w_{t,\delta x})\| \delta\right] \leq \delta L.$$

□

6.2.3 Regret Bound for DBGD

Thus far, we have focused on proving properties regarding the relative loss functions ϵ_t and $\hat{\epsilon}_t$. We can easily bound our regret formulation (6.2) using ϵ_t.

Lemma 14. *Fix $\delta > 0$. Expected regret is bounded by*

$$\mathbf{E}\left[R_T\right] \leq -2\mathbf{E}\left[\sum_{t=1}^{T} \epsilon_t(w^*)\right] + \delta L T.$$

Proof. We can write expected regret as

$$\begin{aligned}
\mathbf{E}\left[R_T\right] &\leq 2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \epsilon^*(w_t)\right] + \delta L T \\
&= -2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \epsilon_t(w^*)\right] + \delta L T
\end{aligned}$$

by noting that $|\epsilon^*(w_t') - \epsilon^*(w_t)| \leq \delta L$, and also that $\epsilon_t(w^*) = -\epsilon^*(w_t)$. □

We now analyze the regret behavior of the smoothed loss functions $\hat{\epsilon}_t$. Lemma 15 provides a useful intermediate result. Note that the regret formulation analyzed in Lemma 15 is different from (6.2).

Lemma 15. *Fix $\delta \in \left(0, \frac{L_\sigma}{L_v L_2}\right)$, and define λ as in (6.11). Assume a sequence of smoothed relative loss functions $\hat{\epsilon}_1, \ldots, \hat{\epsilon}_T$ ($\hat{\epsilon}_{t+1}$ depending on w_t) and $w_1, \ldots, w_T \in \mathcal{W}$ defined by $w_1 = 0$ and $w_{t+1} = \mathbf{P}_{\mathcal{W}}(w_t - \eta g_t)$, where $\eta > 0$ and $g_1, \ldots, g_T$ are vector-valued random variables with (a) $\mathbf{E}[g_t|w_t] = \nabla\hat{\epsilon}_t$, (b) $\|g_t\| \leq G$, and (c) $\mathcal{W} \subseteq R\mathcal{B}$. Then for $\eta = \frac{R}{G\sqrt{T}}$,*

$$\mathbf{E}\left[\sum_{t=1}^{T} \hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)\right] \leq \lambda RG\sqrt{T} + (3+\lambda)\delta T. \tag{6.16}$$

(Adapted from Lemma 3.1 in [65])

Proof. Theorem 6 implies the LHS of (6.16) to be

$$\begin{aligned}
&= \sum_{t=1}^{T} \mathbf{E}\left[\hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)\right] \\
&\leq \sum_{t=1}^{T} \mathbf{E}\left[\ \lambda\nabla\hat{\epsilon}_t(w_t)\cdot(w_t - w^*) + (3+\lambda)\delta L\ \right] \\
&= \lambda\sum_{t=1}^{T} \mathbf{E}\left[\mathbf{E}[g_t|w_t]\cdot(w_t - w^*)\right] + (3+\lambda)\delta LT \\
&= \lambda\sum_{t=1}^{T} \mathbf{E}[g_t\cdot(w_t - w^*)] + (3+\lambda)\delta LT
\end{aligned} \tag{6.17}$$

Following the analysis of [192], we will use the potential function $\|w_t - w^*\|^2$. In particular we can rewrite $\|w_{t+1} - w^*\|^2$ as

$$\begin{aligned}
&= \|\mathbf{P}_{\mathcal{W}}(w_t - \eta g_t) - w^*\|^2 \\
&\leq \|w_t - \eta g_t - w^*\|^2 \\
&= \|w_t - w^*\|^2 + \eta^2\|g_t\|^2 - 2\eta(w_t - w^*)\cdot g_t \\
&\leq \|w_t - w^*\|^2 + \eta^2 G^2 - 2\eta(w_t - w^*)\cdot g_t
\end{aligned} \tag{6.18}$$

where (6.18) follows from the convexity of $\mathcal{W}$. Rearranging terms allows us to bound $g_t \cdot (w_t - w^*)$ as

$$\leq \frac{\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2 + \eta^2 G^2}{2\eta}$$

We can thus bound $\sum_{t=1}^{T} \mathbf{E}[g_t \cdot (w_t - w^*)]$ by

$$\begin{aligned} &\leq \sum_{t=1}^{T} \mathbf{E}\left[\frac{\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2 + \eta^2 G^2}{2\eta}\right] \\ &= \mathbf{E}\left[\frac{\|w_1 - w^*\|^2}{2\eta} + T\frac{\eta^2 G^2}{2\eta}\right] \leq \frac{R^2}{2\eta} + T\frac{\eta G^2}{2} \end{aligned} \tag{6.19}$$

which follows from choosing $w_1 = 0$ and $\mathcal{W} \subseteq R\mathcal{B}$. Combining (6.17) and (6.19) bounds the LHS of (6.16) by

$$\leq \lambda\left(\frac{R^2}{2\eta} + T\frac{\eta G^2}{2}\right) + (3+\lambda)\delta T.$$

Choosing $\eta = \frac{R}{G\sqrt{T}}$ finishes the proof. □

We finally present our main result.

Theorem 7. *By setting* $w_1 = 0$*,*

$$\delta = \frac{\sqrt{2Rd}}{\sqrt{13L}T^{1/4}},\ \gamma = \frac{R}{\sqrt{T}},\ T > \left(\frac{\sqrt{2Rd}L_v L_2}{\sqrt{13L}L_\sigma}\right)^4, \tag{6.20}$$

DBGD achieves expected regret (6.2) bounded by

$$\mathbf{E}\left[R_T\right] \leq 2\lambda_T T^{3/4}\sqrt{26RdL}$$

where

$$\lambda_T = \frac{L_\sigma\sqrt{13L}T^{1/4}}{L_\sigma\sqrt{13L}T^{1/4} - L_v L_2\sqrt{2Rd}}. \tag{6.21}$$

Proof. Adapting from [65], if we let

$$g_t = -\frac{d}{\delta}X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u_t))u_t,$$

using X_t as described in (6.6), then by Lemma 12 and Lemma 13 we have $\mathbf{E}[g_t|w_t] = \nabla\hat{\epsilon}_t(w_t)$. By restricting T in (6.20), we guarantee $\delta \in (0, L_\sigma/L_v L_2)$. We can then apply Lemma 15 using the update rule

$$\begin{aligned} w_{t+1} &= \mathbf{P}_{\mathcal{W}}(w_t - \eta g_t) \\ &= \mathbf{P}_{\mathcal{W}}(w_t + \eta\tfrac{d}{\delta} X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u_t))u_t) \end{aligned}$$

which is exactly the update rule of DBGD if we set $\eta = \gamma\delta/d$. Note that

$$\|g_t\| = \left\| \frac{d}{\delta} X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u_t))u_t \right\| \leq \frac{d}{\delta}.$$

Setting $G = d/\delta$ and noting our choice of $\gamma = R/\sqrt{T}$, we have $\eta = \frac{R}{G\sqrt{T}}$. Applying Lemma 15 yields

$$\mathbf{E}\left[\sum_{t=1}^{T} \hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)\right] \leq \frac{\lambda R d\sqrt{T}}{\delta} + (3+\lambda)\delta LT. \tag{6.22}$$

Combining Lemma 14 and (6.22) yields

$$\begin{aligned} \mathbf{E}[R_T] &\leq -2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \epsilon_t(w^*)\right] + \delta LT \\ &= 2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \epsilon_t(w_t) - \epsilon_t(w^*)\right] + \delta LT \\ &\leq 2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)\right] + 5\delta LT \\ &\leq \tfrac{2\lambda R d\sqrt{T}}{\delta} + (11+2\lambda)\delta LT \\ &\leq \lambda\left(\tfrac{2Rd\sqrt{T}}{\delta} + 13\delta LT\right) \end{aligned}$$

Choosing $\delta = \frac{\sqrt{2Rd}}{\sqrt{13L}T^{1/4}}$ completes the proof. □

Corollary 2. *Using choices of w_1, δ, and γ as stated in Theorem 6.2, if*

$$T > \left(\frac{\sqrt{2Rd}L_v L_2}{\sqrt{13L}L_\sigma}\right)^4 \left(\frac{1+\alpha}{\alpha}\right)^4,$$

for $\alpha > 0$, then

$$\mathbf{E}[R_T] \leq 2(1+\alpha)T^{3/4}\sqrt{26RdL}.$$

The potential non-convexity of $\hat{\epsilon}_t$ significantly complicates the regret bound. By additionally assuming that $\hat{\epsilon}_t$ is convex in W_t (which we have observed empirically in many settings), we arrive at a much simpler result.

Proposition 3. *Assume for all possible w_t that $\hat{\epsilon}_t$ is convex in W_t, which implies*

$$\hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*) \leq \nabla \hat{\epsilon}_t(w_t) \cdot (w_t - w^*).$$

Then for $w_1 = 0$, $\delta = \frac{\sqrt{2Rd}}{\sqrt{5L}T^{1/4}}$, and $\gamma = \frac{R}{\sqrt{T}}$, we have

$$\mathbf{E}[R_T] \leq 2T^{3/4}\sqrt{10RdL}.$$

(Proved as Theorem 9 in Appendix 6)

6.2.4 Practical Considerations

Choosing δ to achieve the regret bound stated in Theorem 7 requires knowledge of ϵ_t (i.e., L), which is typically not known in practical settings. The regret bound is indeed robust to the choice of δ. So sublinear regret is achievable using many choices for δ, as we will verify empirically. In the analysis $w_1 = 0$ was chosen to minimize its distance to any other point in $\mathcal{W}$. In certain settings, we might choose $w_1 \neq 0$, in which case our analysis still follows with slightly worse constants.

6.3 Experiments

6.3.1 Synthetic Simulations

We first experimented using synthetic value functions, which allows us to test the robustness of DBGD to different choices of δ. Since L is unknown, we introduced

Table 6.1: Average regret of DBGD with synthetic functions.

δ_L Factor	0.6	0.8	1	2	3
P_1	0.465	0.398	0.334	**0.303**	0.415
P_2	0.803	0.767	**0.760**	0.780	0.807
P_3	0.687	0.628	**0.604**	0.637	0.663
P_4	0.500	0.378	0.325	**0.304**	0.418
P_5	0.710	**0.663**	0.674	0.798	0.887

a free parameter δ_L and used $\delta = T^{-1/4}\delta_L\sqrt{0.4Rd}$. We tested on five settings P_1 to P_5. Each setting optimizes over a 50-dimensional ball of radius 10, and uses the logistic transfer function with different value functions that explore a range of curvatures (which affects the Lipschitz constant) and symmetries:

$$v_1(w) = -w^T w, \quad v_2(w) = -|w|$$

$$v_3(w) = -\sum_{i:odd} \left(w^{(i)}\right)^2 - \sum_{i:even} \left|w^{(i)}\right|$$

$$v_4(w) = -\sum_i \left[\exp\left(w^{(i)}\right) + \exp\left(-w^{(i)}\right)\right]$$

$$v_5(w) = v_3(w) - \sum_{i:(i\%3=1)} e^{\left[w^{(i)}\right]_+} - \sum_{i:(i\%3=2)} e^{\left[-w^{(i)}\right]_+}$$

The initial point is $w_1 = \mathbf{1}\sqrt{5/d}$. Table 6.1 shows the regret over the interesting range of δ_L values. Performance degrades gracefully beyond this range. Note that the regret of a random point is about 1 since most points in $\mathcal{W}$ have much lower value than $v(w^*)$.

We also compared against Bandit Gradient Descent (BGD) [65]. Like DBGD, BGD explores in random directions at each iteration. However, BGD assumes access to $P(w_t \succ w)$, whereas DBGD only observes random outcomes. Thus BGD assumes strictly more information.[4] We evaluated two versions: BGD1 using

[4]Our analysis yields matching upper bounds on expected regret for all three methods, though it can be shown that the BGD gradient estimates have lower variance.

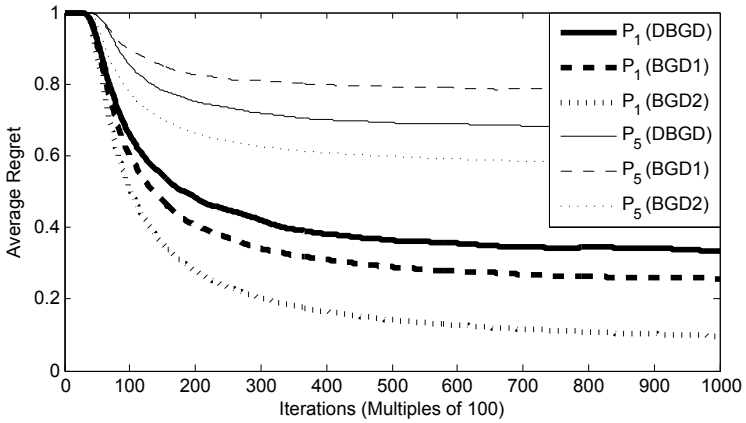

Figure 6.2: Average regret for $\delta_L = 1$

$P(w_t \succ w)$, and BGD2 using $\epsilon_t(w) = P(w_t \succ w) - 1/2$. We expect BGD2 to perform best since the sign of $\epsilon_t(w)$ reveals significant information regarding the true gradient. Figure 6.2 shows the average regret for problems P_1 and P_5 with $\delta_L = 1$. We observe the behaviors of DBGD and BGD being very similar for both. Interestingly, DBGD outperforms BGD1 on P_5 despite having less information. We also observe this trend for P_2 and P_3, noting that all three problems have significant linear components.

6.3.2 Web Search Simulations

For a more realistic simulation environment, we leveraged the same web search dataset that was used in Chapter 5. The idea is to simulate users issuing queries by sampling from queries in the dataset. For each query, the competing retrieval functions will produce rankings, after which the "user" will randomly prefer one ranking over the other; we used a value function based on NDCG@10 (defined below) to determine the comparison outcome probabilities.

We stress that our usage of the dataset is very different from supervised learning

settings. In particular, (extensions of) our algorithm might be applied to experiments involving real users where very little is known about each user's internal value function. We leverage this dataset as a reasonable first step for simulating user behavior in an on-line learning setting.

The training, validation and test sets each consist of 1000 queries. We only simulated on the training set, although we measured performance on the other sets to check for, e.g., generalization power. There are about 50 documents per query, and documents are labeled by 5 levels of relevance from 0 (Bad) to 4 (Perfect). The compatibility between a document/query pair is represented using 367 features. A standard retrieval function computes a score for each document based on these features, with the final ranking resulting from sorting by the scores. For simplicity, we considered only linear functions w, so that the score for document x is $w^T x$. Since only the direction of w matters, we are thus optimizing over a 367-dimensional unit sphere.

Our value function is based on Normalized Discounted Cumulative Gain (NDCG), which is a common measure for evaluating rankings [58]. For query q, NDCG@K of a ranking for documents of q is

$$\frac{1}{N_K^{(q)}} \sum_{k=1}^{K} \frac{2^{r_k} - 1}{\log(k+1)},$$

where r_k is the relevance level of the kth ranked document, and $N_K^{(q)}$ is a normalization factor[5] such that the best ranking achieves NDCG@K=1. For our experiments, we used the logistic function and 10×NDCG@10 to make probabilistic comparisons.

We note a few properties of this setup, some going beyond the assumptions in Section 5.2. This allows us to further examine the generality of DBGD. First,

[5]Note that $N_K^{(q)}$ will be different for different queries.

Table 6.2: Average (upper) and Final (lower) NDCG@10 on Web Search training set (sampling 100 queries/iteration).

$\delta \setminus \gamma$	0.001	0.005	0.01	0.05	0.1
0.5	0.524	0.570	0.580	0.569	0.557
0.8	0.533	0.575	0.582	0.576	0.566
1	0.537	0.575	0.584	0.577	0.568
3	0.529	0.565	0.573	0.575	0.571
0.5	0.559	0.591	0.592	0.569	0.565
0.8	0.564	0.593	0.593	0.574	0.559
1	0.568	0.592	0.595	0.582	0.570
3	0.557	0.581	0.582	0.577	0.576

Table 6.3: Comparing Ranking SVM vs. final DBGD models (with different sampling sizes) using average NDCG@10 and per-query win, tie, and loss counts.

Model	NDCG@10	Win	Tie	Loss
SVM	0.612	–	–	–
Sample 1	0.596	490	121	389
Sample 5	0.593	489	121	390
Sample 10	0.589	504	118	378
Sample 25	0.593	489	118	393
Sample 50	0.596	472	119	409
Sample 100	0.595	490	116	394

the value function is now random (dependent on the query). Second, our feasible space $\mathcal{W}$ is the unit sphere and not convex, although it is a well-behaved manifold. Third, we assume a homogenous user group (i.e., all users have the same value function – NDCG@10). Fourth, rankings vary discontinuously w.r.t. document scores, and NDCG@10 is thus a discontinuous value function. We addressed this issue by comparing multiple queries (i.e., delaying multiple iterations) before an update decision, and also by using larger choices of δ and γ. Lastly, even smoothed versions of NDCG have local optima [58], making it difficult to find w^* (which is required for computing regret). We thus used NDCG@10 to measure performance.

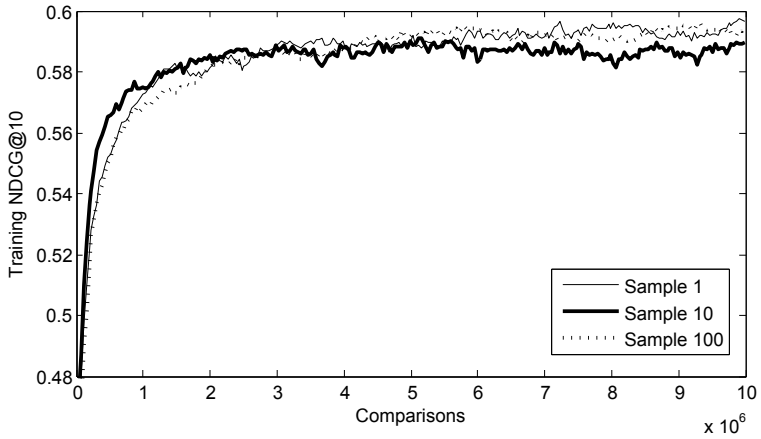

Figure 6.3: NDCG@10 on Web Search training set

We tested DBGD for $T = 10^7$ and a range of γ and δ values. Table 6.2 shows the average (across all iterations) and final training NDCG@10 when comparing 100 queries per update. Performance peaks at $(\delta, \gamma) = (1, 0.01)$ and degrades smoothly. We found similar results when varying the number of queries compared per update. Figure 6.3 depicts per iteration NDCG@10 for the best models when sampling 1, 10 and 100 queries. Making multiple comparisons per update has no impact on performance (the best parameters are typically smaller when sampling fewer queries). Sampling multiple queries is very realistic, since a search system might be constrained to, e.g., making daily updates to their ranking function. Performance on the validation and test sets closely follows training set performance (so we omit their results). This implies that our method is not overfitting.

For completeness, we compared our best DBGD models with a ranking SVM, which optimizes over pairwise document preferences and is a standard baseline in supervised learning to rank settings. More sophisticated methods (e.g., [35, 58]) can further improve performance. Table 6.3 shows that DBGD approaches ranking SVM performance despite making fundamentally different assumptions (e.g., ranking SVMs have access to very specific document-level information). We

caution against over-optimizing here, and advocate instead for developing more realistic experimental settings.

6.4 Discussion

The limitations and research directions discussed in Section 5.6 in the previous chapter, such as extending to unknown time horizons and incorporating shifting user interests, are all applicable to the continuous setting as well. In addition, there also exist other questions specific to the continuum-armed dueling bandits setting. Perhaps the most prominent theoretical question is whether the regret bounds could be improved. The regret bound proved in this chapter is $\mathcal{O}(T^{3/4})$ whereas best known lower bound is $\Omega(\sqrt{T})$ [192]. Recently proposed approaches that achieve nearly tight regret bounds in the standard conntinuum-armed bandit setting [2] may be applicable in the dueling bandits setting as well.

From a practical standpoint, the challenge remains to find the most appropriate modeling assumptions and utility definitions for characterizing these types of on-line interactive learning problems. As was demonstrated in the experiments, Dueling Bandit Gradient Descent (DBGD) is applicable beyond the assumptions stated in Section 5.2, and an algorithm similar to DBGD may be provably efficient for other interactive learning cost models.

CHAPTER 7

INTERPRETING USER FEEDBACK IN INTERLEAVING EXPERIMENTS

When developing interactive algorithms that learn from implicit user feedback, two issues emerge to the foreground. The first is choosing the interaction strategy to optimize for some trade-off between exploration and exploitation; this was discussed in Chapters 5 and 6. The second issue, which we address in this chapter, is how to derive increasingly more useful implicit feedback from observed user interactions.

As was also discussed in previous chapters, one effective approach for deriving reliable judgments from implicit feedback is to focus on collecting *relative* as opposed to *absolute* feedback. For example, while it is difficult to interpret clicks on an absolute scale (e.g., clicked results are relevant, non-clicked results are not relevant), there is clear evidence that clicks provide reliable relative feedback (e.g., clicked results are better than skipped results) [4, 88, 140]. This property is exploited in Interleaving Experiments [83, 140] to compare the relative quality of two ranked retrieval functions h and h'. For every incoming query, the rankings of the two retrieval functions are presented to the user as a single interleaved ranking, and the user's clicks are observed. If the user clicks more on results from h than from h' in the interleaved ranking, it was shown that one can reliably conclude that h is preferred over h' [140, 134]. From an experiment design perspective, interleaving provides a blind paired test where presentation bias is eliminated through randomization under reasonable assumptions.

In this chapter, we aim to make interleaving experiments more efficient – or scalable – by developing a more powerful test statistic. Our motivation comes

from the intuition that not every click in the interleaved ranking is equally informative. For example, a click at rank 1 in a query session immediately followed by a "back" (i.e., a quick return to the search results page) is probably less informative than the last click in the session (which satisfies the information need). As such, having more flexible weighting schemes on clicks can reduce the variance of the test statistic.[1] This improved experiment design will allow us to confidently tease apart the quality of two competing retrieval functions using substantially less data. Note that developing more data-efficient methods for interpreting user feedback can benefit any methodology that relies on such feedback for evaluation and/or optimization, and not just interactive learning approaches.

We present three learning methods for optimizing test statistics by using training data from pairs of retrieval functions of known relative retrieval quality (e.g., by gathering enough data so that the conventional test statistic is significant) [183]. The learned test statistic can then be used to more quickly identify the superior retrieval function in future interleaving experiments. Learning test statistics can be thought of as solving the inverse problem of conventional hypothesis testing, and we present an empirical evaluation on real data from an operational search engine for research papers.

7.1 Interleaving Evaluation

In analogy to experiment designs from sensory analysis (see e.g. [109]), interleaving experiments [83, 140] provide paired preference tests between two retrieval (i.e., ranking) functions. Such paired experiments are particularly suitable in situations where it is difficult or meaningless to assign an absolute rating (e.g., rate

[1] This is also known as the credit assignment problem [134].

Algorithm 8 Team-Draft Interleaving

Input: Rankings $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$
Init: $I \leftarrow (); TeamA \leftarrow \emptyset; TeamB \leftarrow \emptyset;$
while $(\exists i : A[i] \notin I) \wedge (\exists j : B[j] \notin I)$ **do**
  if $(|TeamA| < |TeamB|) \vee$
    $((|TeamA| = |TeamB|) \wedge (RandBit() = 1))$ **then**
    $k \leftarrow \min_i\{i : A[i] \notin I\}$ ..... *top result in A not yet in I*
    $I \leftarrow I + A[k];$ ..... *append it to I*
    $TeamA \leftarrow TeamA \cup \{A[k]\}$ ..... *clicks credited to A*
  else
    $k \leftarrow \min_i\{i : B[i] \notin I\}$ ..... *top result in B not yet in I*
    $I \leftarrow I + B[k]$ ..... *append it to I*
    $TeamB \leftarrow TeamB \cup \{B[k]\}$ ..... *clicks credited to B*
  end if
end while
Output: Interleaved ranking I, $TeamA$, $TeamB$

this taste on a scale from 1 to 10), but a relative comparison is easy to make (e.g., do you like taste A better than taste B). To elicit such pairwise preferences, both alternatives have to be presented side-by-side and without presentation bias. For example, the order in which a subject tastes two products must be randomized, and the identity of the products must be "blind" to the user.

For the case of comparing pairs of retrieval functions, interleaving experiments are designed to provide such a blind and unbiased side-by-side comparison of two retrieval functions h and h'. When a user issues a query q, the rankings $A = h(q)$ and $B = h'(q)$ are computed but kept hidden from the user. Instead, the user is shown a single interleaved ranking I computed from A and B, so that clicks on I provide feedback on the users preference between A and B under reasonable assumptions.

In this chapter, we focus on the Team-Draft Interleaving method [140] that is summarized in Algorithm 8. Team-Draft Interleaving creates a fair (i.e. unbiased) interleaved ranking following the analogy of selecting teams for a friendly team-

	Input Ranking		**Interleaved Rankings** *Team-Draft*			
Rank	A	B	AAA	BAA	ABA	...
1	a	b	a^A	b^B	a^A	
2	b	e	b^B	a^A	b^B	
3	c	a	c^A	c^A	e^B	
4	d	f	e^B	e^B	c^A	
5	g	g	d^A	d^A	d^A	
6	h	h	f^B	f^B	f^B	
⋮	⋮	⋮	⋮	⋮	⋮	

Figure 7.1: An example showing how the Team-Draft method interleaves input rankings A and B for different random coin flip outcomes. Superscripts of the interleavings indicates team membership.

sports match. One common approach is to first select two team captains, who then take turns selecting players in their team. Team-Draft Interleaving uses an adapted version of this approach for creating interleaved rankings. Suppose each document is a player, and rankings A and B are the preference orders of the two team captains. In each round, captains pick the next player by selecting their most preferred player that is still available, add the player to their team and append the player to the interleaved ranking I. We randomize which captain gets to pick first in each round. An illustrative example from [140] is given in Figure 7.1.

To infer whether the user prefers ranking A or ranking B, one counts the number of clicks on documents from each team. If team A gets more clicks, A wins the side-by-side comparison and vice versa. Denoting the sets of clicks on the respective teams with C and C' for query q, the mean or median value of the test statistic

$$\delta(q, C, C') = |C| - |C'| \tag{7.1}$$

over the distribution $P(q)$ of queries reveals whether one of h and h' is consistently preferred over the other. Section 7.1.1 discusses three possible tests that detect whether the mean or median of $\delta(q, C, C')$ is significantly different from zero.

Note that the presentation is unbiased in the sense that A and B have equal probability of occupying each rank in I. This means that any user that clicks randomly will not generate a significant preference in either direction.

In this chapter, we address one shortcoming of the test statistic in (7.1): the test statistic scores all clicks equally, which is likely to be suboptimal in practice. For example, a user clicking back immediately after clicking on a result is probably an indicator that the result was not good after all. The goal here is to learn a more refined function $score(q, c)$ that scores different types of clicks according to their actual information content. This scoring function can then be used in the following rule

$$\delta(q, C, C') = \left[\sum_{c \in C} score(q, c)\right] - \left[\sum_{c' \in C'} score(q, c')\right].$$

Note that this reduces to (7.1) if $score(q, c)$ is always 1.

In the following, we will use a linear model $score(q, c) = w^T \varphi(q, c)$ to score clicks, where w is a vector of parameters to be learned and $\varphi(q, c)$ returns a feature vector describing each click c in the context of the entire query session q. We can now rewrite $\delta(q, C, C')$ as

$$\delta_w(q, C, C') = w^T \Phi(q, C, C')$$

where

$$\Phi(q, C, C') = \sum_{c \in C} \varphi(q, c) - \sum_{c \in C'} \varphi(q, c) \tag{7.2}$$

Feature vectors $\varphi(q, c)$ will contain features that describe the click in relation to position in the interleaved ranking, order and presentation. In Section 7.3.2, we will describe the feature construction used in our empirical evaluation.

7.1.1 Hypothesis Tests for Interleaving

To decide whether an interleaving experiment between h and h' shows a preference in either direction, one needs to test whether some measure of centrality (e.g. median, mean) of the i.i.d. random variables $\Delta_i \equiv \delta(q, C, C')$ is significantly different from zero. For conciseness, let $(\delta_1, ..., \delta_n)$ denote the values of $\delta(q, C, C')$ on a random sample. We consider the following three tests, which will also serve as the baseline methods in our empirical evaluation.

The simplest test, and the one previously used in [83, 84, 140], is the *Binomial Sign Test* (cf. [125]). It counts how often the sign of δ_i is positive, i.e. $S = \sum_{i=1}^{n} [\Delta_i > 0]$. This sum S is a binomial random variable, and the null hypothesis is that the underlying i.i.d. Bernoulli random variables $[\Delta_i > 0]$ have $p = 0.5$.

Unlike the Binomial Sign Test, the *z-Test* (cf. [125]) uses the magnitudes of the Δ_i and tests whether their sum is zero in expectation. The z-Test assumes that $S = \frac{1}{n} \sum_{i=1}^{n} \Delta_i$ is normal, which is approximately satisfied for large n. The ratio of the observed value $s = \frac{1}{n} \sum_{i=1}^{n} \delta_i$ and standard deviation $std(S)$, called the z-score $z = s/std(S)$, monotonically relates to the p-value of the z-test. While $std(S)$ has to be known, an approximate z-test results from estimating $std(S) = \frac{1}{\sqrt{n}} \sqrt{\frac{1}{n} \sum_j (s - \delta_j)^2}$ from the sample. The t-test accounts for the additional variability from the estimate of the standard deviation, but for large samples z-test and t-test are virtually identical.

Finally, we consider the *Wilcoxon Signed Rank Test* (cf. [125]) as a non-parametric test for the median of the Δ_i being 0. To compute the test statistic, the observations are ranked by $|\delta_i|$. Let the resulting rank of δ_i be r_i. The test statistic is then computed as $W = \sum sign(\delta_i) r_i$, and W is tested for mean 0 using

a z-test.

7.2 Learning Methods

The main idea behind learning is to find a scoring function that results in the most sensitive hypothesis test. To illustrate this goal, consider the following hypothetical scenario where the scoring function $score(q, c) = w^T \varphi(q, c)$ differentiates the last click of a query session from other clicks within the same session. The corresponding feature vector $\varphi(q, c)$ would then have two binary features

$$\varphi(q, c) = \begin{pmatrix} \text{1, if c is last click; 0 else} \\ \text{1, if c is not last click; 0 else} \end{pmatrix}.$$

Assume for simplicity that every query session has 3 clicks, with "not last clicks" being completely random while "last clicks" favoring the better retrieval function with 60% probability. Using the weight vector $w^T = (1, 1)$ (i.e., the conventional scoring function), one will eventually identify that the better retrieval function gets more clicks (typically after ≈280 queries using a t-test with $p = 0.95$). However, the optimal weight vector $w^T = (1, 0)$ will identify the better retrieval function much faster (typically after ≈150 queries), since it eliminates noise from the non-informative clicks.

The learning problem can be thought of as an "inverse" hypothesis test: given data for pairs (h, h') of retrieval functions where we know $h \succ h'$, find the weights w that maximizes the power of the test statistic on new pairs. More concretely, we assume that we are given a set of ranking function pairings $\{(h_1, h'_1), ..., (h_k, h'_k)\}$ for which we know w.l.o.g. that h_i is better than h'_i, i.e. $h_i \succ h'_i$. This preference may be known by construction (e.g., h'_i is a degraded version of h_i), by

running interleaving until the conventional test statistic that scores each click uniformly becomes significant, or through some expensive annotation process (e.g., user interviews or manual assessments). For each pair (h_i, h'_i), we assume access to usage logs from Team-Draft Interleaving [140] for n_i queries. For each query q_j, the clicks C_j and C'_j for each "team" are recorded in a triple (q_j, C_j, C'_j). Eventually, all triples are combined into one training sample $S = ((q_1, C_1, C'_1), ..., (q_n, C_n, C'_n))$.[2] After training, the learned w and the resulting test statistic $\delta_w(q, C, C')$ will be applied to new pairs of retrieval functions (h_{test}, h'_{test}) of yet unkown relative retrieval quality.

We now propose three learning methods, with each corresponding to opimizing a specific inverse hypothesis test.

7.2.1 Maximize Mean Difference

In the simplest case, we can optimize the parameters w of $score_w(q, c)$ to maximize the mean difference of scores between the better and the worse retrieval functions,

$$\begin{aligned} w^* &= \operatorname*{argmax}_w \sum_{j=1}^{n} \delta_w(q_j, C_j, C'_j) \\ &= \operatorname*{argmax}_w \sum_{j} w^T \Phi(q_j, C_j, C'_j) \end{aligned}$$

To abstract from different scalings of w and to make the problem well posed, we impose a normalization constraint $||w|| = 1$, leading to the following optimization problem:

$$w^* = \operatorname*{argmax}_w \sum_{j} w^T \Phi(q_j, C_j, C'_j) \text{ s.t. } ||w|| = 1,$$

[2] We are essentially treating all interleaving pairs as a single combined example. A better approach may be to explicitly treat each interleaving pair as a separate example.

which can be written more compactly using $\Psi_j = \Phi(q_j, C_j, C'_j)$,

$$w^* = \underset{w}{\text{argmax}} \left[\sum_j w^T \Psi_j \right] \text{ s.t. } ||w|| = 1.$$

This has the the following closed-form solution that can be derived via Lagrange multipliers:

$$w^* = \frac{\sum_j \Psi_j}{\sqrt{(\sum_j \Psi_j)^T (\sum_j \Psi_j)}} \sim \sum_j \Psi_j.$$

While maximizing the mean difference is intuitively appealing, one key shortcoming is that variance is ignored. In fact, one can think of this method as an inverse z-Test, where we assume equal variance for all w. Since the assumption of equal variance will clearly not be true in practice, we now consider the following more refined methods.

7.2.2 Inverse z-Test

The following learning method removes the assumption of equal variance and optimizes the statistical power of a z-Test in the general case (with the null hypothesis that the mean is zero). Finding the w that maximizes the z-score (and therefore the p-value) on the training set corresponds to the following optimization problem:

$$\begin{aligned} w^* &= \underset{w}{\text{argmax}} \frac{\frac{1}{n} \sum_j \delta_w(q_j, C_j, C'_j)}{\frac{1}{\sqrt{n}} \sqrt{\frac{1}{n} \sum_j \delta_w(q_j, C_j, C'_j)^2 - \left[\frac{1}{n} \sum_j \delta_w(q_j, C_j, C'_j) \right]^2}} \\ &= \underset{w}{\text{argmin}} \frac{\sum_j \delta_w(q_j, C_j, C'_j)^2}{\left[\sum_j \delta_w(q_j, C_j, C'_j) \right]^2} \end{aligned} \tag{7.3}$$

While (7.3) has two symmetric solutions, we are interested only in the one where $\sum_j \delta_{w^*}(q_j, C_j, C'_j) > 0$. Using the abbreviated notation $\Psi_j = \Phi(q_j, C_j, C'_j)$, this

optimization problem can be rewritten as

$$w^* = \underset{w}{\text{argmax}} \frac{(w^T \sum_j \Psi_j)^2}{w^T \left[\sum_j \Psi_j \Psi_j^T\right] w}.$$

For any w solving this optimization problem, cw with $c > 0$ is also a solution. We can thus rewrite the problem as

$$w^* = \underset{w}{\text{argmax}} \left[w^T \sum_j \Psi_j\right] \text{ s.t. } w^T \left[\sum_j \Psi_j \Psi_j^T\right] w = 1.$$

Using the Lagrangian

$$L(w, \alpha) = w^T \sum_j \Psi_j - \alpha \left(w^T \left[\sum_j \Psi_j \Psi_j^T\right] w - 1 \right),$$

and solving for zero derivative w.r.t. w and α, one arrives at a closed form solution. Denoting $\Psi = \sum_j \Psi_j$ and $\Sigma = \sum_j \Psi_j \Psi_j^T$ the solution can be written as

$$w^* = \frac{\Sigma^{-1}\Psi}{\sqrt{\Psi^T \Sigma^{-1} \Psi}}.$$

While not used in our experiments, a regularized version Σ_{reg} of the covariance matrix Σ can be used to prevent overfitting. One straightforward approach is to add a ridge term $\Sigma_{reg} = \Sigma + \gamma I$, where I is the identity matrix and γ is the regularization parameter.

7.2.3 Inverse Rank Test

Last but not least, we consider a learning method that relates to inverting the Wilcoxon Rank Sign test. A good scoring function $\delta_w(q, C, C')$ for the Wilcoxon test should optimize the Wilcoxon statistic, which can be computed as follows. Assuming $h \succ h'$ w.l.o.g., we denote a prediction as "correct" if $\delta_w(q, C, C') > 0$; otherwise, we denote it as incorrect. Ranking all observations by $|\delta_w(q, C, C')|$ (assuming no ties), the Wilcoxon statistic is isomorphic to the number of observation

pairs where an incorrect observation is ranked above a correct observation. One strategy for minimizing the number of such swapped pairs, and therefore optimizing the p-value of the Wilcoxon test, is to choose

$$\delta_w(q, C, C') = \Pr(h \succ h' | q, C, C') - 0.5, \tag{7.4}$$

where $\Pr(h \succ h' | q, C, C')$ is the estimated probability that h is better than h' given the clicks observed for query q.

We estimate $\Pr(h \succ h' | q, C, C')$ from the training data S using a standard logistic regression model

$$\ln \frac{\Pr(h \succ h' | q, C, C')}{\Pr(h' \succ h | q, C, C')} = w^T \Phi(q_j, C_j, C'_j).$$

Using again the convention that $h \succ h'$ for the training data and abbreviating $\Psi_j = \Phi(q_j, C_j, C'_j)$, the parameters w are chosen via maximum likelihood,

$$w^* \quad = \quad \underset{w}{\operatorname{argmax}} \prod_{j=1}^{n} \frac{1}{1 + e^{-w^T \Psi_j}}.$$

w^* denotes the logistic regression solution on the training data. We used the LR-TRIRLS package[3] to solve this optimization problem. The final ranking function can be simplified to the linear function $\delta_w(q, C, C') = w^T \Phi(q, C, C')$, since it produces the same rankings and signs as (7.4).

[3] `http://komarix.org/ac/lr/`

7.3 Empirical Setup

7.3.1 Data Collection

We evaluated our methods empirically using data collected from the Physics E-Print ArXiv.[4] In particular, we used two datasets of click logs collected while running Team-Draft Interleaving experiments. For both datasets, we recorded information for each query (e.g., the entire session) and click (e.g., rank, timestamp, result information, source ranking function, etc). This information is used to generate features for learning (see Section 7.3.2 below). One could also collect user-specific information (e.g., user history), but we have not done so in the following experiments.

"Gold standard". Our first dataset is taken from the Team-Draft experiments described in [140]. In these experiments, the incumbent retrieval function was corrupted in multiple ways to provide pairs of retrieval functions with known relative quality. This provides cheap access to a "gold standard" dataset, since one knows by construction which retrieval function is superior within each pair. A total of six pairs was evaluated, with each yielding slightly over 1000 query sessions.

New interleaving experiments. Our second dataset was generated via interleaving pairs of retrieval functions without necessarily having knowledge of which retrieval function is superior within each pair. For example, one retrieval function we used modifies the incumbant retrieval function by giving additional weight to query/title similarity. It is a priori unclear whether this would result in improved retrieval quality. Ideally (and intuitively), learning a test statistic on the gold stan-

[4] `http://arxiv.org`

dard dataset should help us more quickly determine the superior retrieval function within these interleaving pairs. We examine this hypothesis further in Section 7.4.4. A total of six different retrieval functions are considered in this setting. We collected click data from interleaving every possible pairing of the six, resulting in fifteen interleaving pairs with each yielding between 400 and 650 query sessions. We then removed three of the fifteen interleaving pairs from our analysis, since all methods (including the baselines) showed poor performance (p-value greater than 0.4), making them uninteresting for comparison purposes.

7.3.2 Feature Generation

The features we used describe a diverse set of properties related to clicking behavior, including the rank and order of clicks, and whether search result clicks led to a PDF download in ArXiv. Let C_{own} and C_{other} denote the clicks from the own team and the other team, respectively. Recall from (7.2) that our feature function $\Phi(q, C_{own}, C_{other})$ decomposes as

$$\Phi(q, C_{own}, C_{other}) = \sum_{c \in C_{own}} \varphi(q, c) - \sum_{c \in C_{other}} \varphi(q, c).$$

We will construct $\varphi(q, c)$ for $c \in C_{own}$ in the following way:

1. 1 always
2. 1 if c led to a download
3. $\frac{1}{|C_{own}|}$ if C_{own} gets both more clicks and downloads
4. If $|C_{own}| == |C_{other}|$:
   (a) $\min\left\{\frac{number\_of\_bolded\_words\_in\_title}{number\_of\_query\_words}, 1\right\}$
   (b) $\min\left\{\frac{number\_of\_bolded\_words\_in\_abstract}{number\_of\_query\_words}, 2\right\}$

5. If it is a single-click query:

   (a) 1 if c is not at rank 1

   (b) 1 if c is on first page (top 10)

6. If it is a multi-click query:

   (a) 1 if c is first click

   (b) 1 if c is last click

   (c) 1 if c is first click and not at rank 1

   (d) 1 if c is at rank 1

   (e) 1 if c is at ranks 1 to 3

   (f) 1 if c is on first page (top 10)

   (g) 1 if c is followed by click on a higher position (regression click)

Analogously, we construct $\varphi(q, c)$ for $c \in C_{other}$ by swapping C_{own} and C_{other} in the preceding feature definitions.

Note that some features are more naturally expressed at the query level. For example, feature 3 can be equivalently expressed directly as feature of $\Phi(q, C_{own}, C_{other})$ as

$$\begin{cases} 1 & \text{if } C_{own} \text{ gets both more clicks and downloads} \\ -1 & \text{if } C_{other} \text{ gets both more clicks and downloads} \\ 0 & \text{otherwise} \end{cases}.$$

For clarity, we focus our formulation on click-level features, since most features we used are more naturally understood at the click level.

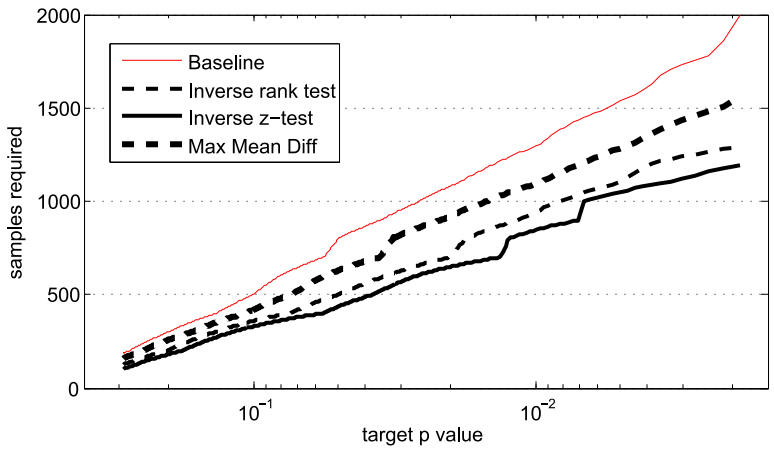

Figure 7.2: Comparing the sample size required versus target t-test p-value in the synthetic experimental setting. Measurements taken from 1000 bootstrapped subsamples for each subsampling size.

7.4 Empirical Evaluation

For ease of presentation, we will only show comparisons against the t-test baseline; our empirical results also hold when comparing against the other baselines. In general, we find the inverse z-test to be the best performing method, with the inverse rank test often being competitive as well.

7.4.1 Synthetic Experiment

We first conducted a synthetic experiment where all six gold standard interleaving pairs in the training set are mixed together to form a single (virtual) interleaving pair. From this, 70% of the data was used for training and the remaining 30% for testing. Intuitively, this setup satisfies the assumption that the click distribution we train on is the same as the click distribution we test on – a core assumption often made when analyzing machine learning approaches.

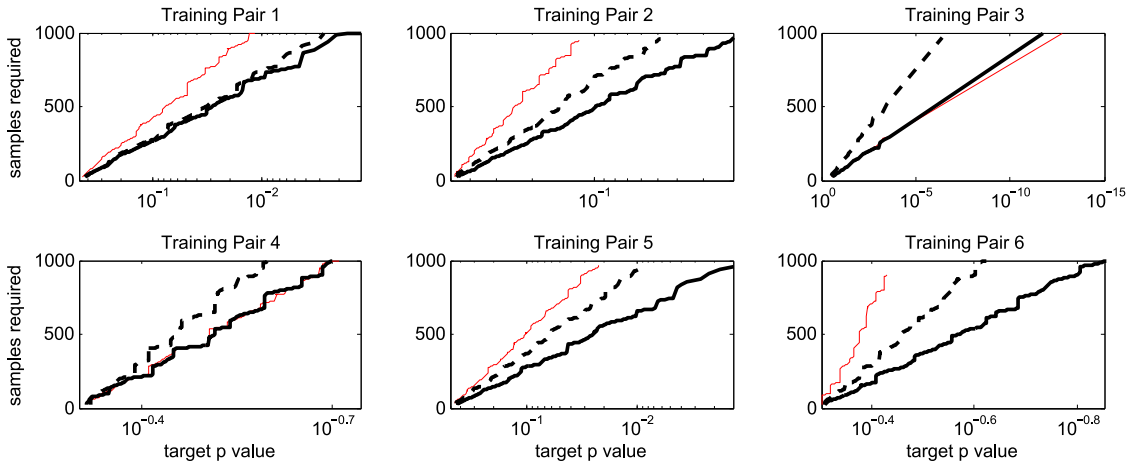

Figure 7.3: Comparing sample size required versus target t-test p-value in leave-one-out testing on the training set. Methods compared are baseline (red), inverse rank test (black dotted) and inverse z-test (black solid). The inverse z-test consistently performs as well as the baseline, and can be much better. Note that the different graphs vary dramatically in scale.

Figure 7.2 shows how the required sample size grows with decreasing target t-test p-value. This plot (and all similar plots) was generated by subsampling the test set (with replacement) at varying subset sizes and computing the p-value. Subset sizes increase in increments of 25 and each subset size was sampled 1000 times. Our goal is to reduce the required sample size, so lower curves indicate superior performance.

We observe in Figure 7.2 that our methods consistently outperform the baseline. For example, for a target p-value of 0.01, the inverse z-test requires only about 800 samples whereas the baseline t-test requires about 1200 – a 50% improvement. In all of our subsequent experiments, we find that the max mean difference method consistently performs worse than the inverse z-test. As such, we will focus on the inverse rank test and the inverse z-test in the remaining empirical evaluations.

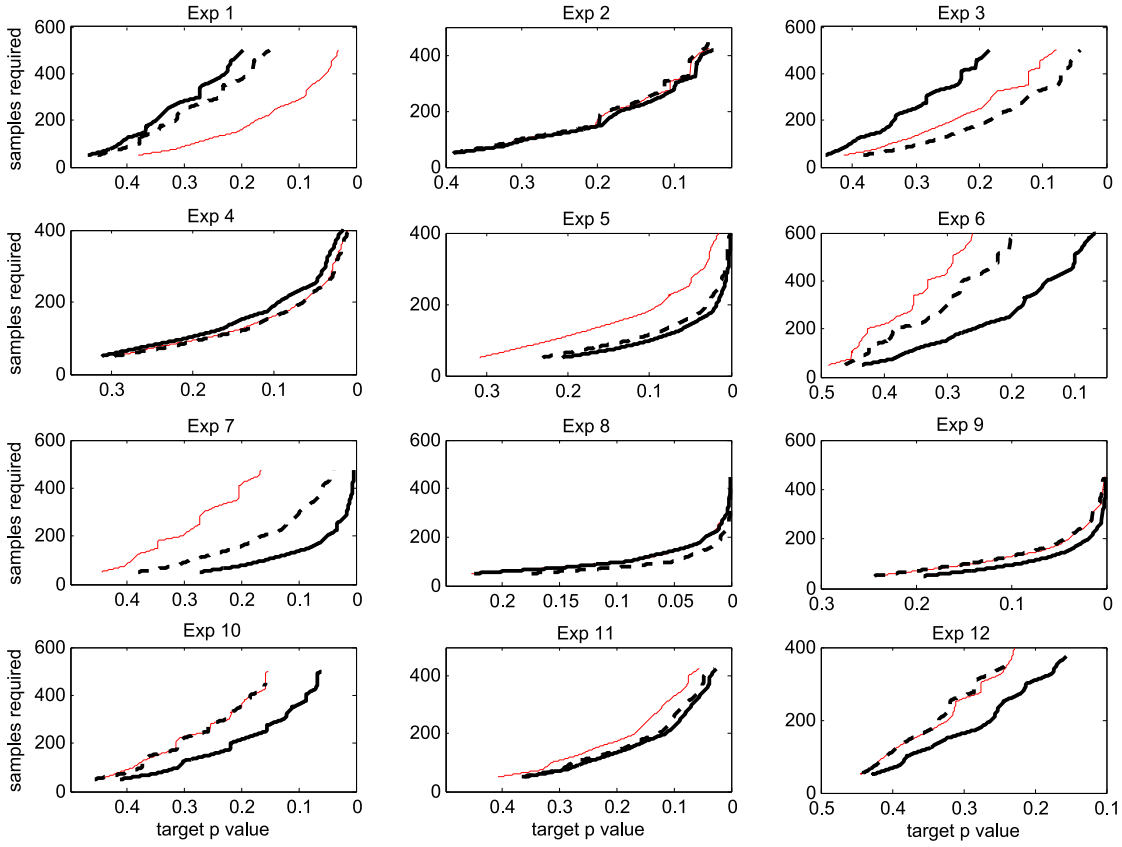

Figure 7.4: Comparing sample size required versus target t-test p-value in the twelve new interleaving experiments. Methods compared are baseline (red), inverse rank test (black dotted) and inverse z-test (black solid). Both the inverse rank test and inverse z-test methods outperform baseline in most cases.

7.4.2 Analyzing the Learned Scoring Function

To give some insight into the scoring function $\delta_w(q, C, C') = w^T \Phi(q, C, C')$ learned by our methods, Table 7.1 shows the weights w generated by the inverse rank test on the full gold standard training set. Since the features are highly correlated, it is difficult to gain insight merely through inspection of the weights. We therefore provide some prototypical example queries for which we will compute the feature vector $\Psi = \Phi(q, C, C')$ and the value of $\delta_w(q, C, C')$.

Table 7.1: Weights learned by the inverse rank test on the full gold standard training set. See Section 7.3.2 for a full description of the features.

ID	Feature Description (w.r.t. $\varphi(q, c)$)	Weight
1	Click	0.056693
2	Download	0.020917
3	More clicks & downloads than other team	0.052410
4a	$\mathbf{1}$[# Clicks equal] $\times$ Title bold frac	0.083463
4b	$\mathbf{1}$[# Clicks equal] $\times$ Abstract bold frac	0.118568
5a	Single click query AND Rank > 1	0.149682
5b	Single click query AND Rank ≤ 10	0.004950
6a	Multi-clicks AND First click	0.063423
6b	Multi-clicks AND Last click	0.000303
6c	Multi-clicks AND First click AND Rank > 1	0.015217
6d	Multi-clicks AND Click at rank $= 1$	0.018800
6e	Multi-clicks AND Click at ranks ≤ 3	-0.00419
6f	Multi-clicks AND Click at ranks ≤ 10	0.067362
6g	Multi-clicks AND Regression click	0.033067

1. *Single click on result from h at rank 1*: Feature vector Ψ has value 1 for features 1 and 5b, leading to $\delta_w = 0.062$ (we are assuming no downloads in this scenario).
2. *Single click on result from h at rank 3*: Feature vector Ψ has value 1 for features 1, 5a and 5b, leading to $\delta_w = 0.211$. As expected, this query is judged to be more informative, since a click at rank 3 indicates a more careful selection.
3. *Single click on result from h at rank 3 followed by download*: Feature vector Ψ has value 1 for features 1, 2, 3, 5a, 5b, leading to $\delta_w = 0.285$. The download adds further evidence, which follows our intuition.
4. *One click on result from h at rank 1, followed by another click on result from h' at rank 2. Rank 2 has bolded title terms, while rank 1 has not*: Feature vector Ψ has value 1 for features 6a, 6d, and value -1 for 4a and 6b. This leads to $\delta_w = -0.002$, indicating a slight preference for h'.

7.4.3 Cross Validation Experiments

In this setting, we trained our models on five of the gold standard interleaving pairs and tested on the remaining one, repeating this process for all six pairs. This provides a controlled way of evaluating generalization performance. Figure 7.3 shows how required sample size changes as the target p-value decreases. Again, lower curves indicate superior performance. We observe the inverse z-test performing at least as well as the baseline on all except training pair 3. Note, however, that training pair 3 is an exceptionally easy case where one can achieve confident p-values with very little data. We observe the inverse rank test to also be competitve, but with somewhat worse performance.

7.4.4 New Interleaving Experiments

To further evaluate the methods in a typical application scenario, we trained our models on all six of the gold standard interleaving pairs, and then tested their predictions on new interleaving pairs. It should be noted that we did not examine the new interleaving dataset when developing the features described in Section 7.3.2. As such, this evaluation very closely matches how such methods would be used in practice.

Figure 7.4 shows, for all twelve test cases, how required sample size changes as the target t-test p-value decreases. We observe both learning methods consistently performing at least as well as, and often much better than, the baseline t-test (with the exception of Exp 1). We also verified that all methods and baselines agree on the direction of the preference in all cases (since we are using a two-tailed test).

Table 7.2: Sample size requirements of three target t-test p-values for the twelve new interleaving experiments.

	Baseline			Inv. rank test			Inv. z-test		
	p=0.2	p=0.1	p=0.05	p=0.2	p=0.1	p=0.05	p=0.2	p=0.1	p=0.05
Exp 1	160	288	406	373	> 500	> 500	491	> 500	> 500
Exp 2	169	310	> 450	149	313	> 450	146	275	416
Exp 3	247	460	> 500	180	330	471	461	> 500	> 500
Exp 4	93	161	228	90	160	230	104	189	251
Exp 5	111	182	259	64	114	162	53	97	142
Exp 6	> 625	> 625	> 625	575	> 625	> 625	254	505	> 625
Exp 7	415	> 475	> 475	157	296	423	76	137	199
Exp 8	59	95	142	< 50	74	99	58	95	144
Exp 9	70	128	174	71	129	184	< 50	94	138
Exp 10	352	> 500	> 500	353	> 500	> 500	216	361	> 500
Exp 11	174	328	> 425	141	260	365	134	222	339
Exp 12	> 400	> 400	> 400	> 400	> 400	> 400	308	> 400	> 400

Table 7.2 provides numerical comparisons for several standard significance thresholds. For half of the twelve test cases, the inverse z-test reduces the required sample size by at least 10% for a target significance of $p = 0.1$. For a quarter of the cases, the inverse z-test achieves a significance of $p = 0.05$ using the available data whereas the baseline t-test fails to do so. These results imply that substantial savings can be gained from employing optimized test statistics.

7.5 Discussion

In this section, we discuss and summarize the core assumptions and limitations of this approach.

While the learned test statistics generally improved the power of the experiments on new retrieval function pairs (h, h'), there is likely a limit to how different

the new pair may be from the training pairs. If the retrieval functions to be evaluated move far from the training data (e.g. after several iterations of improving the ranking function), it might be necessary to add appropriate training data and re-optimize the test statistic. Furthermore, we do not believe that test statistics learned on one search engine would necessarily generalize to a different collection or user population.

A key issue in generalizing to new retrieval function pairs (h, h') lies in the appropriate choice of features $\Phi(q, C, C')$. In particular, if the chosen features allow the learning algorithm to models specific idiosyncracies of the training pairs, this will likely result in poor generalization on new pairs.

Pooling the training examples from multiple training pairs (h_i, h'_j) into one joint training set might lead to unwanted results, since the learning methods optimize an "average" statistic over multiple pairs. In particular, the methods might ignore difficult to discriminate pairs in return for increased discriminative power on easy pairs. It would be more robust to minimize the maximum p-value uniformly over all training pairs.

Finally, the empirical results need to be verified in other retrieval domains. Particularly interesting are domains that include spam. It would be interesting to see whether one can learn scoring functions that recognize (and discount) clicks that were attracted by spam.

7.5.1 Closing the Loop with Interactive Learning

The methods presented in this chapter are complementary to the interactive learning approaches for the Dueling Bandits Problem presented in Chapters 5 and 6.

In the Dueling Bandits Problem, the comparison oracle is assumed to satisfy certain somewhat idealistic properties. From the perspective taken in this chapter, the most important such assumption is that the comparison oracle should exactly reflect the distribution of user preferences. In other words, the probability of retrieval function A winning a comparison versus retrieval function B is assumed to exactly reflect the degree of preference for A versus B. What is not known are the relative qualities of the retrieval functions – this gives rise to the regret cost model in the Dueling Bandits Problem.

In this chapter we consider a setting that is opposite to what was considered in the Dueling Bandits Problem. Here, we assume that the comparison oracle is noisy and does not perfectly reflect user preferences, but we instead have prior knowledge of relative retrieval quality. Furthermore, this noise also tends to be "biased" due to presentation effects of the comparison oracle. This leads to an interesting "chicken versus egg" dilemma when trying to combine this approach with the Dueling Bandits Problem into a unified setting, and a solution could potentially have significant practical as well as theoretical value.

Part IV

Conclusion and Appendices

CHAPTER 8

CONCLUSION AND OUTLOOK

Managing digital information is a growing problem in every application domain, ranging from integrating biological data, browsing digital libraries, organizing personal content, searching in specialty domains, or filtering news feeds and Twitter updates. Over the past 20 years, learning to rank approaches have proven to be invaluable with their ability to combine coarse human feedback with statistical regularities of the prediction domain in order to derive effective models. This has enabled the development of a wide variety of intelligent information systems, and their effectiveness is evidenced by their widespread commercial adoption.

But existing approaches use coarse and relatively unrepresentative models of user utility. Thus, applying conventional machine learning approaches often results in optimizing the wrong criteria. And although these surrogate criteria that existing approaches optimize for are typically somewhat aligned with true user utility (which is a major reason for their practical successes thus far), properly applying these techniques can be quite labor intensive and requires substantial hands-on expertise. This inherently limits the scope and reasoning power of the systems that we can efficiently deploy today.

The ultimate goal of any information system is to optimize user utility, and a good general model of user utility should be efficiently and accurately adaptable to any target domain. This requires us to move away from conventional approaches that rely on expensive manual judgments that are necessarily ignorant of user context, and towards methods that can learn rich, structured models from feedback collected via user interaction.

Following the above intuition, this dissertation has proposed methods to address the following two challenges to applying learning to rank methods more broadly and to greater effectiveness:

- Learning to optimize more sophisticated models of user utility
- Learning to interact with users and collect more representative feedback

And while the approaches proposed herein are by no means the complete and final solutions to these challenges, they represent real progress towards a more unified learning framework for designing increasingly intelligent information systems. A particularly salient feature in all the approaches presented herein is that they address these challenges by identifying a fundamental issue of practical importance, and thus motivate models which directly tackle deep and cross-cutting research questions.

The contributions of this dissertation include (1) methods for developing **structured prediction models** that can accomodate rich models of user utility (such as models for maximizing diversity), (2) an **interactive learning framework** for modeling system/user interactions that leads to a well-founded trade-off between exploration and exploition, and (3) methods for deriving more useful feedback from observed user interactions (i.e., implicit feedback).

The structured prediction approaches described in Chapters 3 and 4 show how we can move beyond simple hypothesis classes and models of user utility that make very unrealistic independence assumptions, thus addressing the first challenge stated above. For example, Chapter 4 describes a structured prediction approach that can learn to perform well in diversified retrieval settings, where retrieving redundant documents (something not well-modeled by previous learn-

ing approaches) can significantly degrade retrieval quality. Previous methods for tackling diversified retrieval typically apply a sequence of filters of clustering algorithms in addition to using a conventional retrieval function. When applying such approaches, it is often difficult to state – much less automatically optimize – the learning objective or user utility function, and thus they require significant "hand-holding" by human experts in order to achieve some benefit.

The on-line learning framework described in Chapters 5 and 6, called the Dueling Bandits Problem, provides a simple yet practical reformulation of the conventional multi-armed bandits setting that leverages the growing body of methods designed to elicit *relative* as opposed to *absolute* feedback. For example, the Interleaved Filter algorithms described in Chapter 5 provide a way to automatically schedule on-line interleaving experiments using a pool of thousands or millions of candidate retrieval functions. Each interleaving experiment is a blind on-line test that yields noisy information regarding the relative quality of two retrieval functions. This is done by showing users an interleaving of the two rankings and observing clicks (e.g., more clicks on A or B?). But each on-line experiment also incur a cost due to potentially decreasing user utility (the two rankings that were interleaved might be very poor). The Dueling Bandits Problem directly models this exploration/exploitation dilemma using a suitable notion of regret, and the Interleaved Filter 2 algorithm was proven to be information-theoretically optimal (up to constant factors).

Finally, Chapter 7 presents methods for deriving more informative feedback from observed use behavior, such as clicks collected via on-line interleaving experiments. It is well known that implicit feedback, though plentiful, is often very noisy and biased, and deriving more useful (i.e., less noisy and/or biased) implicit feed-

back is a subject of intense study. But most such methods typically tackle the issue somewhat indirectly from a modeling perspective, such as by making very strong assumptions about user behavior or utility. In contrast, the methods proposed in Chapter 7 directly learn to optimize the efficiency of various statistical hypothesis tests that are typically used when evaluating the relative quality of competing retrieval functions. Designing more effective methods to elicit implicit feedback is an important complementary research direction to the interactive learning problem explored in Chapters 5 and 6.

8.1 Future Directions

In this final section, we discuss how related research fields can also benefit from the methods proposed in this dissertation, as well as more general information retrieval problems. All of these problems can be tackled by developing more powerful structured prediction and interactive learning approaches.

There are many parallels between the fields of Information Retrieval and Natural Language Processing (NLP). Both deal primarily with text, which leads to very similar modeling requirements. Thus, one expects many of the learning techniques developed to also be applicable for problems in NLP. For example, as discussed in Section 4.6, the document summarization task can be well modeled as a coverage problem [52], and the $\text{SVM}^{\Delta}_{div}$ method presented in Chapter 4 directly learns coverage models that optimizes for coverage-based utility functions.

For many domains including and beyond NLP, such as Computer Vision and Computational Biology, it is becoming clear that a major limiting factor in current approaches is that model development is often done separate from the usage con-

texts (e.g., providing a ranking of recommendations). This immediately suggests rich and potentially fruitful directions of future research at the intersection of Information Retrieval and many other domains, and the learning problems embedded therein.

Fundamentally, interactive learning models how systems can learn to maximize (the users') utility through the actions they choose to take. This can be applied to numerous domains. For example, given an appropriate comparison mechanism, one can apply frameworks similar to the Dueling Bandits Problem to model computer-assisted teaching (by interactively learning the best teaching strategy for some population of students), product recommendation, providing driving directions, computer-assisted scheduling, and much more.

To make developing such applications feasible, it appears necessary to rely heavily (perhaps completely) on implicit feedback derived from observed user behavior. This is an attractive approach since observed user behavior (e.g., clicks on search results, movement patterns tracked by cell phones, or behavioral patterns observed in "smart" homes) is both cheap to collect and naturally representative of the target user population (e.g., web users, individuals, or family-sized groups). As discussed earlier, developing effective methods for inferring implicit feedback is a vital complementary problem to interactive learning, and is also much more application dependent.

Further research on modeling user interactions can also provide new methodologies for analyzing our numerous digital social networks. Currently, two methodologies exist: link (cf. [128, 96]) and text analysis (cf. [120, 154]). Usage data is, in a sense, more democratic since it reflects the preferences of the end users rather than the content creators. In many domains, content creators are also end users (e.g.,

centralized scholarly libraries such as ArXiv), leading to an interesting symbiosis between information access systems and the evolution of digital social networks. In addition, studying user interactions can lead to a richer understanding of how topics and concepts flow through a digital corpus. For example, by examining co-click data, one might tease apart subtopics (from the users' perspective) to generate feedback for learning retrieval models in the aforementioned diversified retrieval setting.

As our society becomes more data-driven, applications and tasks of all types will come to increasingly rely upon information systems. Stated differently, information services can potentially aid us in every aspect of our lives, even those that are not currently viewed as being information constrained. But our growing number of systems and services are becoming ever more difficult to maintain and configure. This line of research can lead to cost-effective information systems that can efficiently adapt to variety of retrieval domains such as enterprise search, library search, medical search, and the many new and exciting applications to come.

APPENDIX A

SUPPLEMENTARY MATERIAL FOR CHAPTER 4

A.1 Maximizing Coverage

Let U denote the universe of elements to be covered. Each element $u \in U$ is associated with a non-negative weight $w(u)$. Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ denote a collection of sets, where each $B_k \subset U$ "covers" a subset of the elements in U. We write the benefit of covering a subset of the universe $V \subset U$ as

$$F(V) = \sum_{u \in V} w(u). \tag{A.1}$$

The goal then is to select a subcollection $Y \subset \mathcal{B}$ of size K that maximally covers U, or

$$Y = \underset{Y' \subset \mathcal{B}, |Y'| \leq K}{\operatorname{argmax}} F(U(Y')), \tag{A.2}$$

where

$$U(Y) = \bigcup_{B_k \in Y} B_k.$$

We will show that the naive greedy algorithm described in Algorithm 9 achieves a $1 - 1/e$ approximation guarantee of optimal. Note that this optimization problem generalizes the coverage problems discribed in Chapter 4 and that Algorithm 9 is essentially equivalent to Algorithm 3.

The optimization problem in (A.2) is an instance of the budgeted maximum coverage problem [95, 79]. We now prove the $1 - 1/e$ approximation bound based on the analysis technique presented in [79].

Algorithm 9 Greedy selection by myopically maximizing weighted coverage

1: Input: U,B,K
2: Initialize solution $A_0 \leftarrow \emptyset$
3: **for** $k = 1, \ldots, K$ **do**
4: $\quad \hat{B} \leftarrow \text{argmax}_{B:B \notin A_{k-1}} \{F(U(A_{k-1}) \cup B) - F(U(A_{k-1}))\}$
5: $\quad A_k \leftarrow A_{k-1} \cup \{\hat{B}\}$
6: **end for**
7: **return** A_K

Let OPT_K denote the value of the optimal solution. Let $A_1, \ldots, A_K$ denote the sequence of (partial) solutions generated by the greedy algorithm at end of each iteration, and let $a_1, \ldots, a_K$ denote the value of the greedy solution at each iteration, i.e.

$$a_k = F(U(A_k)).$$

Note that a_K is the value of the greedy solution. We also define

$$a(k) = \sum_{j \leq k} a_j.$$

In the following analysis, we consider a more general setting where each iteration of the greedy algorithm might not necessarily find the maximally beneficial set at each iteration, and instead finds a set $\hat{B}$ that is a β-approximation to the maximum weight set available. In other words, in Line 4 in Algorithm 9, the solution $\hat{B}$ at each iteration k satisfies

$$\sum_{u \in U(A_{k-1}) \cup \hat{B}} w(u) - \sum_{u \in U(A_{k-1})} w(u) \geq \beta \max_{B:B \notin A_{k-1}} \left\{ \sum_{u \in U(A_{k-1}) \cup B} w(u) - \sum_{u \in U(A_{k-1})} w(u) \right\}.$$

Lemma 16. *For $k = 1, 2, \ldots, K$, we have*

$$a_k \geq \frac{\beta}{K} \left(OPT_K - a_{k-1} \right).$$

Proof. At least $OPT_K - a_{k-1}$ worth of elements not covered by A_k are covered by the by the K sets in the optimal solution. Hence, by the pigeonhole principle, one

of the K sets in the optimal solution must cover at least $(OPT_K - A_{k-1})/K$ worth of these elements. Since Algorithm 9 finds a set that is a β-approximation of the maximum weight set available, the result follows. □

Lemma 17. *For $k = 1, 2, \ldots, K$, we have*

$$a(k) \geq \left(1 - \left(1 - \frac{\beta}{K}\right)^k\right) OPT_K.$$

Proof. We prove by induction on k. The base case $k = 1$ follows immediately from Lemma 16. For the inductive case, we have

$$\begin{aligned}
a(k+1) &= a(k) + a_{k+1} \\
&\geq a(k) + \frac{\beta}{K}\left(OPT_K - a(k)\right) \\
&= \left(1 - \frac{\beta}{k}\right) a(k) + \frac{\beta}{K} OPT_K \\
&\geq \left(1 - \frac{\beta}{K}\right)\left(1 - \left(1 - \frac{\beta}{K}\right)^k\right) OPT_K + \frac{\beta}{K} OPT_K \\
&= \left(1 - \left(1 - \frac{\beta}{K}\right)^{k+1}\right) OPT_K,
\end{aligned}$$

where the first inequality follows from Lemma 16, and the second inequality follows from the induction hypothesis. □

We now state the main result.

Theorem 8. *Using the notation defined above, we have*

$$a(K) \geq \left(1 - \left(1 - \frac{\beta}{K}\right)^K\right) OPT_K \geq \left(1 - \frac{1}{e^\beta}\right) OPT_K.$$

Proof. The proof follows immediately from Lemma 17 and observing that $(1 - \beta/K)^K$ approaches $1/e^\beta$ from below as $K \to \infty$. □

Corollary 3. *Running Algorithm 9 with size input K returns a prediction A_K that has value*

$$a_k \geq \left(1 - \frac{1}{e}\right) OPT_K.$$

The utility function F defined in (A.1) is an instance of a *submodular* utility function. The submodularity property states that for sets $S \subset S'$

$$F(S \cup s) - F(S) \geq F(S' \cup s) - F(S'),$$

which can be interpreted as characterizing a notion of diminishing returns. It can be shown that for a large class of submodular functions, the greedy algorithm described in Algorithm 9 achieves a $1 - 1/e$ approximation bound, and that this bound is tight in the worst case [127, 95, 103].

APPENDIX B

SUPPLEMENTARY MATERIAL FOR CHAPTER 5

B.1 Satisfying Modeling Assumptions

The following lemma describes a general family of probabilistic comparison models and proves that both strong stochastic transitivity and stochastic triangle inequality are satisfied by this family of models. Note that both the logistic and Gaussian models described in Section 5.2 are contained within this family of models.

Lemma 18. *Let each bandit $b_i \in \{b_1 \dots b_K\}$ be associated with a distinct real value μ_i such that outcomes from comparing two bandits are determined by*

$$P(b_i > b_j) = \sigma(\mu_i - \mu_j),$$

for some transfer function σ. Let σ satisfy the following properties:

- *σ is monotonically increasing*
- $\sigma(-\infty) = 0$
- $\sigma(\infty) = 1$
- $\sigma(x) = 1 - \sigma(-x)$ *(rotation symmetric)*
- *$\sigma(x)$ has a single inflection point at $\sigma(0) = 1/2$*

Then these probabilistic comparisons satisfy strong stochastic transitivity and stochastic triangle inequality.

Proof. We begin by noting that that these properties essentially mean that σ behaves like a symmetric cumulative distribution function with a single inflection point at $\sigma(0) = 1/2$ (i.e., σ is an "S-shaped" curve).

For any triplet of bandits $b_i \succ b_j \succ b_k$, we know that $\mu_i > \mu_j > \mu_k$. To show strong stochastic transitivity, we first note that σ is monotonically increasing. Thus we know that $\sigma(\mu_i - \mu_k) \geq \sigma(\mu_i - \mu_j)$ and $\sigma(\mu_i - \mu_k) \geq \sigma(\mu_j - \mu_k)$, which implies that

$$\begin{aligned}
\epsilon_{i,k} &= \sigma(\mu_i - \mu_k) - \frac{1}{2} \\
&\geq \max\left\{\sigma(\mu_i - \mu_j) - \frac{1}{2}, \sigma(\mu_j - \mu_k) - \frac{1}{2}\right\} \\
&= \max\left\{\epsilon_{i,j}, \epsilon_{j,k}\right\}.
\end{aligned}$$

To show stochastic triangle inequality, we first note that $\sigma(x)$ is sub-additive, or concave, for $x \geq 0$. Define

$$\alpha = \frac{\mu_i - \mu_j}{\mu_i - \mu_k}$$

such that $(\mu_i - \mu_j) = \alpha(\mu_i - \mu_k)$ and $(\mu_j - \mu_k) = (1 - \alpha)(\mu_i - \mu_k)$. Then we know from concavity of σ that

$$\alpha\sigma(\mu_i - \mu_k) + (1 - \alpha)\sigma(0) \leq \sigma(\mu_i - \mu_j),$$

and also

$$(1 - \alpha)\sigma(\mu_i - \mu_k) + \alpha\sigma(0) \leq \sigma(\mu_j - \mu_k).$$

Adding the two inequalities above yields

$$\sigma(\mu_i - \mu_k) + \mu(0) \leq \sigma(\mu_i - \mu_j) + \sigma(\mu_j - \mu_k),$$

and thus

$$\epsilon_{i,k} \leq \epsilon_{i,j} + \epsilon_{j,k}.$$

□

B.2 Analyzing the Random Walk Model

We first describe a family of measure spaces which will be used to analyze the coupling between executions of IF and the Random Walk Model described in Definition 1.

Definition 2. *We define a family of measure spaces $\mathcal{M}$ in the following way. Each point in the sample space is a joint realization of the sequences of random variables X_{ij}^{rt} and Z_i^r for every pair of bandits b_i and b_j, and positive integers r and t. We will define a joint distribution over the random variables X_{ij}^{rt} and a conditional distribution over the Z_i^r variables given the X_{ij}^{rt} variables. The random variables and their distributions are explained in greater detail below.*

- *For every pair of bandits b_i, b_j, and positive integer r, there is a sequence of Bernoulli random variables X_{ij}^{rt} (for $t = 1, 2, \ldots$) describing the outcomes of comparisons in a match played by b_i and b_j in round r provided that b_i is the incumbent in that round. In particular $X_{ij}^{rt} = 1$ if b_i wins the t-th comparison between b_j in round r, and $X_{ij}^{rt} = 0$ if b_i loses that comparison. We will also define the following useful notation to denote prior execution histories: $\mathcal{X}_i^r$ is the σ-field generated by the random variables $\{X_{ij}^{qt} : \ j \neq i, q < r, \ t = 1, 2, \ldots\}$.*
- *For a fixed i, the random variables X_{ij}^{rt} are all mutually independent as one varies j, r, t, and they have the correct distribution for each pair i, j. (In other words, the probability of b_i beating b_j is $1/2 + \epsilon_{ij}$).*
- *For convenience we also define Y^r, for every positive integer r, to denote the identity of the incumbent in round $r + 1$ (i.e., the bandit that wins round r) when running algorithm IF with the comparison outcomes specified by $\{X_{ij}^{rt}\}$.*

Note that the value (likewise distribution) of Y^r is completely determined by the values (joint distribution) of X_{ij}^{rt}.

- *For every bandit b_i and positive integer r, there is a random variable Z_i^r taking non-negative integer values, such that the distribution of $Y^r + Z_i^r$, conditioned on $\mathcal{X}_i^r$, is uniform on $1, \ldots, i-1$ at every sample point where $Y^{r-1} \leq i$ and IF does not make a mistake in rounds $1, \ldots, r$. (This will later be used to show that the Random Walk Model stochastically dominates any mistake-free execution of IF.)*

The values of X_{ij}^{rt} completely determine the history of execution of IF.[1] Our independence assumptions ensure that the history of play observed by IF has the correct distribution over histories.

A priori, it is not obvious that measure spaces $\mathcal{M}$ satisfying Definition 2 exist; the constraint on the conditional distribution of $Y^r + Z_i^r$ is non-trivial but we prove below that it is possible to design a measure space that satisfies this constraint, i.e. $\mathcal{M}$ is not empty. We will then show how any measure space in $\mathcal{M}$ defines a stochastic coupling between the number of rounds required in mistake-free executions of IF and the length of random walks in the Random Walk Model. To begin proving that $\mathcal{M}$ is non-empty, we first prove a constraint on the distribution of the Y^r variables.

Lemma 19. *For any measure space in $\mathcal{M}$, we have*

$$\forall r,\ \forall j \in \{1, 2, \ldots, i-1\} :\ \sum_{j'=1}^{j} P(Y^r = j' | \mathcal{X}_i^r, N^r) \geq \frac{j}{i-1}, \tag{B.1}$$

[1] Some of the values X_{ij}^{rt} are exposed as IF runs and schedules matches. Other values never get exposed. In particular, for pairs of bandits b_i and b_j where neither is the incumbent in round r, the values X_{ij}^{rt}have no bearing on the history of play observed by IF.

where b_i denotes the incumbent bandit chosen by IF for round r, the Y^r and X_{ij}^{rt} variables and the $\mathcal{X}_i^r$ σ-field are defined as in Definition 2, and N^r denotes the event that IF does not make a mistake in round r.

Proof. We will prove the following inequality,

$$\forall t \geq t_{min}, \forall r,\ \forall j \in \{1, 2, \ldots, i-1\} :\ \sum_{j'=1}^{j} P(Y^r = j' | \mathcal{X}_i^r, N^{rti}) \geq \frac{j}{i-1}, \quad \text{(B.2)}$$

where t_{min} denotes the minimum number of comparisons required for IF to determine a winner, and N^{rti} denotes the event that IF does not make a mistake in round r, that b_i is the incumbent in that round, and that IF makes exactly t comparisons between b_i and each other remaining bandit in round r. Since (B.2) will be shown to apply for all feasible t, i, then (B.1) will also hold.

It suffices to show that

$$\forall 1 \leq j < k < i :\ P(Y^r = j | \mathcal{X}_i^r, N^{rti}) \geq P(Y^r = k | \mathcal{X}_i^r, N^{rti}), \quad \text{(B.3)}$$

since then (B.2) follows from iteratively applying the pigeonhole principle (for $j = 1, \ldots, i-1$), and noting that

$$\sum_{j'=1}^{i-1} P(Y^r = j' | \mathcal{X}_i^r, N^{rti}) = 1.$$

Let $U(i, k, r, t | \mathcal{X}_i^r)$ denote the collection of comparison sequences of length t in round r between the incumbent b_i and each other remaining b_j which results in b_k being declared the winner after t comparisons. In other words, an element in $U(i, k, r, t | \mathcal{X}_i^r)$ consists of a realization of each $X_{ij}^{t'r}$ for incumbent b_i, all remaining b_j, and time steps $1 \leq t' \leq t$. It is straightforward to see that

$$P(Y^r = k | \mathcal{X}_i^r, N^{rti}) = P(U(i, k, r, t | \mathcal{X}_i^r) | \mathcal{X}_i^r, N^{rti}).$$

We define a bijection between $U(i,j,r,t|\mathcal{X}_i^r)$ and $U(i,k,r,t|\mathcal{X}_i^r)$ for $j<k$ such that $P(U(i,j,r,t|\mathcal{X}_i^r)|\mathcal{X}_i^r,N_{rt}) \geq P(U(i,k,r,t|\mathcal{X}_i^r)|\mathcal{X}_i^r,N^{rti})$, which directly implies (B.3). Each $u_k \in U(i,k,r,t|\mathcal{X}_i^r,N^{rti})$ is mapped to the corresponding point $u_j \in U(i,j,r,t|\mathcal{X}_i^r,N^{rti})$ that consists of the same sequences of comparisons as u_k, except that the comparison sequences involving b_j and b_k are swapped (implying that b_j is declared the winner).

It remains to show that $P(u_j|\mathcal{X}_i^r,N^{rti}) \geq P(u_k|\mathcal{X}_i^r,N^{rti})$ for all u_j,u_k pairings in the bijection. In the sequences of comparisons defined by u_k, let

$$A=\sum_{t'=1}^{t} X_{ik}^{rt'} \quad \text{and} \quad B=\sum_{t'=1}^{t} X_{ij}^{rt'},$$

where $A>B$. Under the corresponding u_j, the two summations are reversed,

$$B=\sum_{t'=1}^{t} X_{ik}^{rt'} \quad \text{and} \quad A=\sum_{t'=1}^{t} X_{ij}^{rt'},$$

and all other sequences of variables $X_{ii'}^{rt'}$ for $i'\neq j$, $i'\neq k$ remain the same. We also know that $P(X_{ik}^{rt}) \leq P(X_{ij}^{rt})$, since b_k is inferior to b_j. Let $p=P(X_{ij}^{rt})$ and $q=P(X_{ik}^{rt})$. Since all the $X_{ii'}^{rt'}$ variables are mutually independent, we can write the ratio of the conditional probabilities of u_j and u_k as

$$\begin{aligned}
\frac{P(u_j|\mathcal{X}_i^r,N^{rti})}{P(u_k|\mathcal{X}_i^r,N^{rti})} &= \frac{P(\sum_{t=1}^{t'} X_{ij}^{rt}=A)P(\sum_{t=1}^{t'} X_{ik}^{rt}=B)}{P(\sum_{t=1}^{t'} X_{ij}^{rt}=B)P(\sum_{t=1}^{t'} X_{ik}^{rt}=A)} \\
&= \frac{p^A(1-p)^{t'-A}q^B(1-q)^{t'-B}}{p^B(1-p)^{t'-B}q^A(1-q)^{t'-A}} \\
&= \frac{p^{A-B}(1-q)^{A-B}}{q^{A-B}(1-p)^{A-B}} \geq 1
\end{aligned}$$

where the first equality follows from noting that all comparisons are independent and canceling out common terms (i.e., the realizations of $X_{ii'}^{rt}$ for $i'\neq j$ and $i'\neq k$), and the last inequality follows from noting that $A>B$ and $p>q$.

□

Corollary 4. *For the setting described in Lemma 19, we also have*

$$\forall r,\ \forall 1 \leq j < i:\ \sum_{j'=1}^{j} P(Y^r = j' | \mathcal{X}_i^r, N^r) \geq \frac{j}{i'-1},$$

where $i' \geq i$.

Lemma 20. *The family of measure spaces* $\mathcal{M}$ *defined in Definition 2 is non-empty.*

Proof. We will use the notation for $X_{jk}^{rt}, Y^r, Z_j^r, \mathcal{X}_j^r$ as described in Definition 2. We will show that it is possible to construct a distribution on the non-negative random variables Z_j^r which satisfies the requirements of Definition 2. Since we are conditioning on X_{ij}^{qt} for all $q < r$, then the value of Y^{r-1} is fixed (i.e., we know who the incumbent is in round r). Assume WLOG that $Y^{r-1} = i$ (i.e., the incumbent in round r is b_i). We will construct Z_i^r based on the following two cases.

Case 1: IF does not make a mistake in round r and $Y^{r-1} \leq i$ (meaning the incumbent during round r was b_i). We will use the following flow network to construct the conditional distribution of $Y^r + Z_i^r$ (given $\mathcal{X}_i^r$ and N^r),

- source s and sink t
- vertices $u_1, \ldots, u_{i-1}$
- vertices $v_1, \ldots, v_{i-1}$
- edges from s to each u_j with capacity $P(Y^r = j | \mathcal{X}_i^r, N^r)$
- edges from each u_j to v_k where $k \geq j$ with infinite capacity
- edges from each v_k to t with capacity $1/(i-1)$

Lemma 19 and Corollary 4 imply that the minimum s-t cut of this network has capacity 1, and consequently the maximum s-t flow has value 1. In any maximum

flow, each edge (s, u_j) and each edge (v_j, t) (for $1 \leq j \leq i-1$) must be saturated. Given a maximum flow, we can interpret the flow on the edge from u_j to v_k to be the joint conditional probability $P(Y^r = j,\ Z_i^r = k - j \mid \mathcal{X}_i^r, N^r)$, from which we can recover the conditional distribution of Z_i^r given $\mathcal{X}_i^r$ and N^r. The fact that the conditional distribution of $Y^r + Z_i^r$ is uniform on $1, \ldots, i-1$, given $\mathcal{X}_i^r, N^r$, follows from the fact that the flow from v_k to t is exactly $1/(i-1)$ for every k.

Case 2: IF does make a mistake in round r or $Y^{r-1} > i$. Then we set Z_i^r to some arbitrary non-negative integer, e.g., 0.

Thus, we have shown that there exists a feasible probability distribution on the Z_i^r variables which satisfies the requirements of Definition 2, which implies that $\mathcal{M}$ is non-empty. □

Lemma 21. *There exists a stochastic coupling between IF and the Random Walk Model such that the number of rounds in mistake-free executions of IF is stochastically dominated by the length of random walks in the Random Walk Model.*

Proof. We can take any measure space in $\mathcal{M}$ to construct our stochastic coupling, and we know from Lemma 20 that at least one such measure space exists. There is one sample point for every possible joint outcome of the random variables X_{ij}^{rt} and Z_i^r. The execution of IF is determined by the X_{ij}^{rt} variables. Consider any execution of IF that is mistake-free through rounds $1, \ldots, s$. The analogous execution of the Random Walk Model is determined by looking at the sequence of incumbents when one runs a "perturbed" version of IF. The perturbation consists to taking the identity of the incumbent in round $r+1$ (for every $r = 1, \ldots, s$) and modifying it by adding Z_i^r (where b_i is the incumbent of "perturbed" IF in round r), and then executing round $r+1$ using the perturbed incumbent instead of the one that

would ordinarily be chosen by IF. Both IF and "perturbed" IF start with the same initial incumbent at the beginning of round 1 chosen uniformly from $1, \ldots, K$.

Let b^r and $\tilde{b}^r$ be the incumbents chosen by IF and the analogous "perturbed" IF, respectively, at round r (note that $b^r = b_{i'}$ where $i' = Y^{r-1}$). Then it suffices to show that any mistake-free execution of IF satisfies $b^r \succeq \tilde{b}^r$ for all $r > 0$. It is straightforward to see that this stochastic coupling holds from the definition of the Y^r and Z_i^r variables in Definition 2, so long the initial condition $b^1 \succeq \tilde{b}^1$ holds (and $b^1 = \tilde{b}^1$ by definition). $\square$

APPENDIX C

SUPPLEMENTARY MATERIAL FOR CHAPTER 6

C.1 A Simpler Regret Analysis Using Stronger Convexity Assumptions

In this section, we assume for all possible points $w_t \in \mathcal{W}$ that $\hat{\epsilon}_t$ is convex in the region $W_t = \{w : v(w) \geq v(w_t)\}$. Recall that

$$\hat{\epsilon}_t(w) = \mathbf{E}_{x \in \mathcal{B}}[\epsilon_t(\mathbf{P}_{\mathcal{W}}(w + \delta x))],$$

and that

$$\epsilon_t(w) \equiv \epsilon(w_t, w).$$

Using this assumption, we first prove a simpler version of Lemma 15 that does not require Theorem 6.

Lemma 22. *Assume a sequence of smoothed relative loss functions $\hat{\epsilon}_1, \ldots, \hat{\epsilon}_T$ ($\hat{\epsilon}_{t+1}$ depending on w_t and convex in W_t) and $w_1, \ldots, w_T \in \mathcal{W}$ defined by $w_1 = 0$ and $w_{t+1} = \mathbf{P}_{\mathcal{W}}(w_t - \eta g_t)$, where $\eta > 0$ and $g_1, \ldots, g_T$ are vector-valued random variables with (a) $\mathbf{E}[g_t|w_t] = \nabla\hat{\epsilon}_t$, (b) $\|g_t\| \leq G$, and (c) $\mathcal{W} \subseteq R\mathcal{B}$. Then for $\eta = \frac{R}{G\sqrt{T}}$,*

$$\mathbf{E}\left[\sum_{t=1}^{T} \hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)\right] \leq RG\sqrt{T}. \tag{C.1}$$

(Adapted from Lemma 3.1 in [65])

Proof. Since $\hat{\epsilon}_t$ is convex in W_t, then we know that the LHS of (C.1) can be written

as

$$\mathbf{E}\left[\sum_{t=1}^{T}\hat{\epsilon}_t(w_t)-\hat{\epsilon}_t(w^*)\right] \leq \sum_{t=1}^{T}\mathbf{E}\left[\nabla\hat{\epsilon}_t(w_t)\cdot(w_t-w^*)\right]$$
$$=\sum_{t=1}^{T}\mathbf{E}\left[\mathbf{E}[g_t|w_t]\cdot(w_t-w^*)\right]$$
$$=\sum_{t=1}^{T}\mathbf{E}[g_t\cdot(w_t-w^*)] \tag{C.2}$$

Following the analysis of [192], we will use the potential function $\|w_t-w^*\|^2$. In particular we can rewrite $\|w_{t+1}-w^*\|^2$ as

$$\begin{aligned}\|w_{t+1}-w^*\|^2 &= \|\mathbf{P}_{\mathcal{W}}(w_t-\eta g_t)-w^*\|^2\\ &\leq \|w_t-\eta g_t-w^*\|^2 \qquad \text{(C.3)}\\ &= \|w_t-w^*\|^2+\eta^2\|g_t\|^2-2\eta(w_t-w^*)\cdot g_t\\ &\leq \|w_t-w^*\|^2+\eta^2G^2-2\eta(w_t-w^*)\cdot g_t\end{aligned}$$

where (C.3) follows from the convexity of $\mathcal{W}$. Rearranging terms allows us to bound $g_t\cdot(w_t-w^*)$ as

$$g_t\cdot(w_t-w^*)\leq\frac{\|w_t-w^*\|^2-\|w_{t+1}-w^*\|^2+\eta^2G^2}{2\eta}$$

We can thus bound $\sum_{t=1}^{T}\mathbf{E}[g_t\cdot(w_t-w^*)]$ by

$$\sum_{t=1}^{T}\mathbf{E}[g_t\cdot(w_t-w^*)] \leq \sum_{t=1}^{T}\mathbf{E}\left[\frac{\|w_t-w^*\|^2-\|w_{t+1}-w^*\|^2+\eta^2G^2}{2\eta}\right]$$
$$=\mathbf{E}\left[\frac{\|w_1-w^*\|^2}{2\eta}+T\frac{\eta^2G^2}{2\eta}\right]\leq\frac{R^2}{2\eta}+T\frac{\eta G^2}{2} \tag{C.4}$$

which follows from choosing $w_1=0$ and $\mathcal{W}\subseteq R\mathcal{B}$. Combining (C.2) and (C.4) bounds the LHS of (C.1) by

$$\mathbf{E}\left[\sum_{t=1}^{T}\hat{\epsilon}_t(w_t)-\hat{\epsilon}_t(w^*)\right]\leq\frac{R^2}{2\eta}+T\frac{\eta G^2}{2}.$$

Choosing $\eta=\frac{R}{G\sqrt{T}}$ finishes the proof. □

Theorem 9. *Assume for all possible w_t that $\hat{\epsilon}_t$ is convex in W_t, which implies*

$$\hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*) \leq \nabla \hat{\epsilon}_t(w_t) \cdot (w_t - w^*).$$

Then for $w_1 = 0$, $\delta = \frac{\sqrt{2Rd}}{\sqrt{5L}T^{1/4}}$, and $\gamma = \frac{R}{\sqrt{T}}$, we have

$$\mathbf{E}[R_T] \leq 2T^{3/4}\sqrt{10RdL}.$$

Proof. Adapting from [65], if we let

$$g_t = -\frac{d}{\delta} X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u_t))u_t,$$

using X_t as described in (6.6), then by Lemma 12 and Lemma 13 we have $\mathbf{E}[g_t|w_t] = \nabla \hat{\epsilon}_t(w_t)$. We can then apply Lemma 22 using the update rule

$$\begin{aligned} w_{t+1} &= \mathbf{P}_{\mathcal{W}}(w_t - \eta g_t) \\ &= \mathbf{P}_{\mathcal{W}}(w_t + \eta \tfrac{d}{\delta} X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u_t))u_t) \end{aligned}$$

which is exactly the update rule of DBGD if we set $\eta = \gamma\delta/d$. Note that

$$\|g_t\| = \left\| \frac{d}{\delta} X_t(\mathbf{P}_{\mathcal{W}}(w_t + \delta u_t))u_t \right\| \leq \frac{d}{\delta}.$$

Setting $G = d/\delta$ and noting our choice of $\gamma = R/\sqrt{T}$, we have $\eta = \frac{R}{G\sqrt{T}}$. Applying Lemma 22 yields

$$\mathbf{E}\left[\sum_{t=1}^{T} \hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)\right] \leq \frac{Rd\sqrt{T}}{\delta}. \tag{C.5}$$

Combining Lemma 14 and (C.5) yields

$$\begin{aligned} \mathbf{E}[R_T] &\leq -2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \epsilon_t(w^*)\right] + \delta LT \\ &= 2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \epsilon_t(w_t) - \epsilon_t(w^*)\right] + \delta LT \\ &\leq 2\mathbf{E}\left[\textstyle\sum_{t=1}^{T} \hat{\epsilon}_t(w_t) - \hat{\epsilon}_t(w^*)\right] + 5\delta LT \\ &\leq \tfrac{2Rd\sqrt{T}}{\delta} + 5\delta LT \end{aligned}$$

Choosing $\delta = \frac{\sqrt{2Rd}}{\sqrt{5L}T^{1/4}}$ completes the proof.

□

BIBLIOGRAPHY

[1] M. Adler, P. Gemmell, M. Harchol-Balter, R. Karp, and C. Kenyon. Selection in the presence of noise: The design of playoff systems. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1994.

[2] A. Agarwal, O. Dekel, and L. Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2010.

[3] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[4] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[5] N. Ailon and M. Mohri. An efficient reduction of ranking to classification. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2008.

[6] A. Al-Maskari, M. Sanderson, and P. Clough. The relationship between ir effectiveness measures and user satisfaction. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[7] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

[8] G. Andrew. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

[9] J. Arguello, J. Elsas, J. Callan, and J. Carbonell. Document representation and query expansion models for blog recommendation. In *Proceedings of the Conference of the American Association for Artificial Intelligence (AAAI)*, 2008.

[10] J.-Y. Audibert and S. Bubeck. Minimax policies for adversarial and stochastic bandits. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2009.

[11] P. Auer. Using confidence bounds for exploitation-exploration trade. *Journal of Machine Learning Research (JMLR)*, 3:397–422, 2003.

[12] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[13] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[14] M. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. Sorkin. Robust reductions from ranking to classification. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2007.

[15] M. Ben-Or and A. Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *Processing of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

[16] A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, 2009.

[17] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[18] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[19] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, 2003.

[20] J. Boyan, D. Freitag, and T. Joachims. A machine learning architecture for optimizing web search engines. In *AAAI Workshop on Internet Based Information Systems*, 1996.

[21] L. Breiman, J. Friedman, C. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall, 1993.

[22] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[23] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *Proceedings of the Text REtrieval Conference (TREC)*, 1994.

[24] C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010.

[25] C. Burges, R. Ragno, and Q. Le. Learning to rank with non-smooth cost functions. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2006.

[26] C. Burges, T. Sheked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, Bonn, Germany, Aug. 2005.

[27] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, 2004.

[28] G. Cao, J.-Y. Nie, J. gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.

[29] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[30] Z. Cao, T. Qin, T.-Y. Liu, M.-F. T, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2007.

[31] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and reproducing summaries. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 1998.

[32] B. Carterette and D. Petkova. Learning a ranking from pairwise preferences.

In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[33] R. Caruana, S. Baluja, and T. Mitchell. Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 1996.

[34] N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Regret minimization under partial monitoring. *Mathematics of Operations Research*, 31(3):562–580, 2006.

[35] S. Chakrabarti, R. Khanna, U. Sawant, and C. Battacharyya. Structured Learning for Non-Smooth Ranking Losses. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

[36] M.-W. Chang, V. Srikumar, D. Goldwasser, and D. Roth. Structured output learning with indirect supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

[37] O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS workshop on Machine Learning for Web Search*, 2007.

[38] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval Journal*, 13(3):201–215, 2010.

[39] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the World Wide Web Conference (WWW)*, 2009.

[40] H. Chen and D. Karger. Less is more: Probabilistic models for retrieving fewer relevant documents. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[41] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

[42] W. Chu and S. Keerthi. New approaches to support vector ordinal regression. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[43] C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.

[44] W. Cohen, R. Schapire, and Y. Singer. Learning to order things. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 1998.

[45] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[46] M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. Journal of Machine Learning Research (to appear).

[47] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. J. Wiley, 1999.

[48] M. Cramer, M. Wertheim, and D. Hardtke. Demonstration of improved search result relevancy using real-time implicit relevance feedback. In *SIGIR Workshop on understanding the User – Logging and Interpreting User Interactions in Information Search and Retrieval*, 2009.

[49] K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2001.

[50] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *ACM Conference on Web Search and Data Mining (WSDM)*, 2008.

[51] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. A framework for selective query expansion. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2008.

[52] H. T. Dang. Overview of duc 2006. In *Proceedings of DUC 2006*, 2006.

[53] H. T. Dang, J. Lin, and D. Kelly. Overview of the trec 2006 question an-

swering track. In *Proceedings of the Text REtrieval Conference (TREC)*, 2006.

[54] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[55] M. desJardins, E. Eaton, and K. Wagstaff. Learning user preferences for sets of objects. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

[56] F. Diaz and D. Metzler. Improving the estimation of relevance models using large external corpora. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[57] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(103):103–137, 1997.

[58] P. Donmez, K. Svore, and C. Burges. On the Local Optimality of LambdaRank. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2009.

[59] S. Dumais and H. Chen. Hierarchical Classification of Web Content. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2000.

[60] E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research (JMLR)*, 7:1079–1105, 2006.

[61] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5), 1994.

[62] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[63] T. Finley and T. Joachims. Supervised k-means clustering. Technical Report 1813-11621, Cornell University, 2008.

[64] T. Finley and T. Joachims. Training structural svms when exact inference

is intractable. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[65] A. Flaxman, A. Kalai, and H. B. McMahan. Online Convex Optimization in the Bandit Setting: Gradient Descent Without a Gradient. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

[66] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2004.

[67] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS: Journal of Computer and System Sciences*, 55, 1997.

[68] Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 1998.

[69] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[70] S. I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

[71] L. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www-search. In *Proceedings of the World Wide Web Conference (WWW)*, 2004.

[72] J. Guiver and E. Snelson. Learning to rank with softrank and gaussian processes. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.

[73] D. Hawking. Overview of the TREC-9 web track. In *Proceedings of TREC-2000*, 2000.

[74] D. Hawking and N. Craswell. Overview of the TREC-2001 web track. In *Proceedings of TREC-2001*, Nov. 2001.

[75] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries

for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[76] A. Herschtal and B. Raskutti. Optimising area under the roc curve using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[77] W. Hersh and P. Over. Trec-8 interactive track report. In *Proceedings of the Text REtrieval Conference (TREC)*, 1999.

[78] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[79] D. Hochbaum and A. Pathria. Analysis of th greedy approach in problems of maximum k-coverage. *Naval Research Logistics*, 45:615–627, 1998.

[80] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[81] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[82] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2000.

[83] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[84] T. Joachims. Evaluating retrieval performance using clickthrough data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*, pages 79–96. Physica/Springer Verlag, 2003.

[85] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[86] T. Joachims. Training linear SVMs in linear time. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, 2006.

[87] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[88] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), 2005.

[89] T. Joachims, T. Hofmann, Y. Yue, and C.-N. Yu. Predicting structured objects with support vector machines. *Communications of the ACM, Research Highlight*, 52(11):97–104, November 2009.

[90] K. S. Jones, S. Walker, and S. Robertson. Ambiguous requests: Implications for retrieval tests. *SIGIR Forum*, 41(2):8–17, 2007.

[91] R. M. Karp and R. Kleinberg. Noisy binary search and its applications. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[92] G. Kazai, M. Lalmas, and A. de Vries. The overlap problem in content-oriented xml retrieval evaluation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2004.

[93] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. *ACM SIGIR Forum*, 37(2):18–28, 2003.

[94] C. Kemp and K. Ramamohanarao. Long-term learning for web search engines. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2002.

[95] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1997.

[96] J. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM: Journal of the ACM*, 46, 1999.

[97] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.

[98] R. Kleinberg. Nearly tight bounds for the continuum-armed bandit prob-

lem. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2004.

[99] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2008.

[100] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the Symposium on Theory of Computation (STOC)*, 2008.

[101] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(2):147–159, 2004.

[102] S. Kramer, G. Widmer, B. Pfahringer, and M. D. Groeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, XXI:1001–1013, 2001.

[103] A. Krause and C. Guestrin. A note on the budgeted maximization of submodular functions. Technical Report CMU-CALD-05-103, Carnegie Mellon University, 2005.

[104] O. Kurland and L. Lee. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[105] O. Kurland and L. Lee. Respect my authority! HITS without hyperlinks, utilizing cluster-based language models. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[106] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[107] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[108] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[109] D. Laming. *Sensory Analysis*. Academic Press, 1986.

[110] X. Lan, S. Roth, D. Huttenlocher, and M. Black. Efficient belief propagation with learned higher-order markov random fields. In *Proposition of the European Conference on Computer Vision (ECCV)*, 2006.

[111] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2007.

[112] T. Lau and E. Horvitz. Patterns of search: Analyzing and modeling web query refinement. In *Proceedings of the International Conference on User Modeling*, 1999.

[113] P. Li, C. Burges, and Q. Wu. Learning to rank using classification and gradient boosting. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2007.

[114] Y. Li, W. Luk, K. Ho, and F. Chung. Improving weak ad-hoc queries using wikipedia as external corpus. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[115] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop: Text Summarization Branches Out*, 2004.

[116] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2010.

[117] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[118] T.-Y. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7(1):36–43, 2005.

[119] P. Long and R. Servedio. Boosting the area under the roc curve. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2007.

[120] G. Mann, D. Mimno, and A. McCallum. Bibliometric impact measures leveraging topic analysis. In *Proceedings of the ACM Joint Conference on Digital Libraries (JCDL)*, 2006.

[121] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[122] S. Mannor and J. N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research (JMLR)*, 5:623–648, 2004.

[123] D. Metzler and B. Croft. A markov random field for term dependencies. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[124] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[125] A. Mood, F. Graybill, and D. Boes. *Introduction to the Theory of Statistics*. McGraw-Hill, 1974.

[126] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[127] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[128] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report SIDL-WP-1999-0120, Stanford University, Nov. 1999.

[129] S. Pandey, D. Agarwal, D. Chakrabarti, and V. Josifovski. Bandits for taxonomies: A model-based approach. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, 2007.

[130] J. Ponte. *Language Models for Relevance Feedback*, pages 73–96. Springer, 2000.

[131] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 1998.

[132] T. Qin, X. Hang, D. Wang, T.-Y. Liu, W. Lai, and H. Li. Ranking with multiple hyperplanes. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[133] A. Quattoni, M. Collins, and T. Darrell. Conditional random field for object recognition. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2004.

[134] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2010.

[135] F. Radlinski and T. Joachims. Evaluating the Robustness of Learning from Implicit Feedback. In *ICML Workshop on Learning In Web Search*, 2005.

[136] F. Radlinski and T. Joachims. Query Chains: Learning Rank from Implicit Feedback. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.

[137] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

[138] F. Radlinski and T. Joachims. Active Exploration for Learning Rankings from Clickthrough Data. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2007.

[139] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[140] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Conference on Information and Knowledge Management (CIKM)*, 2008.

[141] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[142] G. Rätsch and S. Sonnenburg. Large scale hidden semi-markov svms. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2006.

[143] S. Robertson. *The Probability Ranking Principle in IR*, pages 281–286. Morgan Kaufmann Publishers Inc., 1997.

[144] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of TREC-3*, 1994.

[145] S. Robertson and H. Zaragoza. On rank-based effectiveness measures and optimisation. Technical report, Microsoft Research, 2006.

[146] J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. Prentice Hall, 1971.

[147] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[148] S. Roth and M. Black. Field of experts: A framework for learning image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[149] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145, 2003.

[150] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[151] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society of Information Science*, 41(4):288–297, 1990.

[152] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, November 1975.

[153] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[154] B. Shaparenko and T. Joachims. Information genealogy: Uncovering the flow of ideas in non-hyperlinked document databases. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2007.

[155] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. Analysis of a very large altavista query log. Technical Report 1998-014, Digital SRC, 1998.

[156] A. Swaminathan, C. Mathew, and D. Kirovski. Essential pages. Technical Report MSR-TR-2008-015, Microsoft Research, 2008.

[157] Q. Tan, X. Chai, W. Ng, and D. Lee. Applying co-training to clickthrough data for search engine adaptation. In *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA)*, 2004.

[158] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[159] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[160] B. Taskar, C. Guestrin, and D. Koller. Max margin markov networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2003.

[161] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

[162] B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research (JMLR)*, 7:1627–1653, 2006.

[163] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: Optimizing nonsmooth rank metrics. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, 2008.

[164] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning (ICML)*, 2004.

[165] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6(Sep):1453–1484, 2005.

[166] A. Turpin and F. Scholer. User performance versus precision measures for simple search tasts. In *ACM Conference on Information Retrieval (SIGIR)*, 2006.

[167] V. Vapnik. *Statistical Learning Theory*. Wiley and Sons Inc., 1998.

[168] E. M. Vorhees and D. K. Harman, editors. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[169] K. Wang, T. Walker, and Z. Zheng. Pskip: Estimating relevance ranking quality from web search clickthrough data. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2009.

[170] X. Wei and W. B. Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[171] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and hang Li. Listwise approach to learning to rank - theory and algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[172] D. Xing, G. Xue, Q. Yang, and Y. Yu. Deep classifier: Automatically categorizing search results into large-scale hierarchies. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, 2008.

[173] J. Xu and H. Li. A boosting algorithm for information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[174] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W. Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.

[175] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma. Directly optimizing ir evaluation measures in learning to rank. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2008.

[176] J. Yedidia, W. Freeman, and Y. Weiss. Understand belief propagation and its generalizations. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[177] C.-N. Yu, T. Joachims, and R. Elber. Support vector training of protein alignment models. In *Proposition of the Information Conference in Research in Computational Biology (RECOMB)*, 2007.

[178] C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *International Conference on Machine Learning (ICML)*, 2009.

[179] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. Journal of Computer and System Sciences (JCSS), Special Issue on Learning Theory (in submission).

[180] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. In *Conference on Learning Theory (COLT)*, 2009.

[181] Y. Yue and C. Burges. On using simultaneous perturbation stochastic approximation for ir measures; and, the empirical optimality of lambdarank. In *NIPS workshop on Machine Learning for Web Search*, 2007.

[182] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[183] Y. Yue, Y. Gao, O. Chapelle, Y. Zhang, and T. Joachims. Learning more powerful test statistics for click-based retrieval evaluation. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2010.

[184] Y. Yue and T. Joachims. Predicting diverse subsets using structural svms. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2008.

[185] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *International Conference on Machine Learning (ICML)*, 2009.

[186] C. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[187] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2001.

[188] C. Zhai and J. Lafferty. A study of smoothing methods for language models

applied to ad hoc information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[189] B. Zhang, H. Li, Y. Liu, L. Ji, W. Xi, W. Fan, Z. Chen, and W. Ma. Improving web search results using affinity graph. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[190] Z. Zheng, H. Zha, T. Zhang., O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2007.

[191] X. Zhu, A. Goldberg, J. V. Gael, and D. Andrzejewski. Improving diversity in ranking using absorbing random walks. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2007.

[192] M. Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.