BIAS MITIGATION TECHNIQUES AND A COST-AWARE FRAMEWORK FOR BOOSTED RANKING ALGORITHMS

by

SOPHIE SALOMON

Submitted in partial fulfillment of the requirements

For the degree of Master of Science

Department of Computer and Data Science

CASE WESTERN RESERVE UNIVERSITY

May, 2020

Bias Mitigation Techniques and a Cost-Aware Framework for Boosted Ranking Algorithms

Case Western Reserve University
Case School of Graduate Studies

We hereby approve the thesis[1] of

SOPHIE SALOMON

for the degree of

Master of Science

Dr. Harold Connamacher

Committee Chair, Adviser — April 02, 2020
Department of Computer and Data Science

Dr. Soumya Ray

Committee Member — April 02, 2020
Department of Computer and Data Science

Dr. Mehmet Koyuturk

Committee Member — April 02, 2020
Department of Computer and Data Science

[1]We certify that written approval has been obtained for any proprietary material contained therein.

Dedicated to the friends, professors, and caffeine that made this happen

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my sincere gratitude to the following individuals who have directly contributed to my understanding of and work with researching the Machine Learning ranking problem. First and foremost, I thank my advisor, Professor Harold Connamacher, for initially involving me with this topic, and for his wisdom, patience, and occasional enforcement of deadlines throughout this process. I would also like to thank Professor Soumya Ray, from whose classes I have learned much of what I know about Artificial Intelligence today, and whose insight into Machine Learning research and datasets helped me while I was developing empirical tests for this work. For their work on our group project on ranking in Machine Learning (EECS 440), I also appreciate the efforts of Nicklaus Roach, Bingwen Ma, and I-Kung Hsu. And of course, thank you to Clarinda Ho and Seohyun Jung for being the best “totally coincidental" group I could have hoped for in that class, brilliant academics, and excellent friends besides.

Abstract

Bias Mitigation Techniques and a Cost-Aware Framework for Boosted Ranking Algorithms

Abstract

by

SOPHIE SALOMON

Recent work in bias mitigation has introduced new strategies and metrics for training fairer machine learning classification models. Current research has focused on the problem of binary classification, which has strongly influenced the techniques developed to prevent elements from the protected class from being characterized accordingly. However, extending these approaches to the ranking problem introduces additional nuance. Accordingly, this paper presents a framework for evaluating the efficacy of ranking fairness metrics which shows existing approaches to be inadequate. Furthermore, this paper demonstrates the properties of a flexible cost-aware paradigm for boosted ranking algorithms and discusses the potential extensions for bias mitigation in the ranking problem. The two problems are fundamentally linked by their shared purpose of reducing risk of either costly or unfair decisions by the trained ranker. Included are the experimental results of the cost-aware versions of RankBoost for ranking and multilabel classification datasets, and exploratory experimentation with using cost-sensitive ranking for bias mitigation.

Keywords: Asymmetric Machine Learning, Cost-Awareness, Bias, Fairness, Ranking, RankBoost, Boosting

1 Introduction

As machine learning techniques achieve widespread usage across sensitive applications, work on bias mitigation and cost-aware training is increasingly urgent. Algorithms will learn patterns available in the training data, which can result in undesirable or even discriminatory behavior. Even using unbiased data to train the model does not completely preclude bias from appearing. A fair model should not reflect membership in a *protected class*, i.e. some characteristic such as race or gender which should not influence individual outcomes for a given problem, in its labeling of the data. However, defining fairness metrics is still an ongoing process for binary classification. This chapter includes discussion (1.2.2) of many of the basic considerations for fair classification to provide background for the challenges of achieving fair ranking.

With many different applications for fair machine learning models, it is appropriate to have a spectrum of robust bias mitigation techniques. Though classification theory is often considered sufficient to establish the state-of-the-art for the analogous theory in different areas of machine learning, this paper explicates the specific challenges associated with extending this work to algorithms that solve the ranking problem. The added complexity of completely ordering the set of elements introduces additional risks and constraints which may lead to unfair rankers without specific research into bias mitigation for machine learning ranking algorithms. This desire to train unbiased rankers is inherently linked to the need for theoretically sound cost-sensitive ranking because both fairness and cost are linked through the need to mitigate

risk. Although there is not yet a technique specifically using cost-sensitive ranking to elicit fair ranking (in part because the research on both problems is very sparse so far), cost-sensitivity works to lower the risk of "expensive" mistakes which is analogous to the need to mitigate the risk of systematic unfair treatment of members of a protected class of elements.

This chapter provides the necessary background, with an accompanying literature review, to understand the original work described in the subsequent chapters on biased ranking and cost-sensitive variants of boosted ranking algorithms. First, 1.1 provides an introduction to the ranking problem and explains basic terminology for ranking, including an overview of boosting. Then, 1.2 gives a survey of work done in bias and fairness for classification, including an overview of fairness metrics and approaches in 1.2.2. This introduction to current work in fair ML for classification is the foundation for Chapter 2, which challenges recent attempts to apply these metrics to fair ranking problems and provides a framework for evaluating new metrics. Finally, this introduction concludes with a section on asymmetric machine learning (1.3) which is essential to understand the work on cost-sensitive ranking in Chapter 3. The original contributions of this paper are to

- Introduce a framework by which to evaluate bias metrics for ranking by defining desirable properties for such metrics.
- Demonstrate that existing extensions of classification bias metrics, used for current research in the area, have vital shortcomings according to this framework.
- Create a cost-sensitive approach for boosted ranking algorithms and prove how applying cost before, during, and after training results in the same loss function.
- Prove the properties of the resulting cost-sensitive RankBoost and RankBoost+ variants, including a generalization bound.
- Discuss how existing performance metrics can reflect cost-sensitive ranking problems and suggest a new cost-sensitive discounting metric CSDCG.

- Provide experimental results which give a proof-of-concept for cost-sensitive RankBoost and RankBoost+ by investigating how cost-sensitivity changes the trained model.

1.1 The Ranking Problem

The ranking problem encompasses many different applications where the goal is to train a model that orders elements based on their features. As opposed to classification, which seeks to label elements into distinct sets, ranking gives each element a relative position[2]. Ranking algorithms train on elements with labeled ranks to create a model that can then order new elements according to the learned patterns. In some cases this can be thought of as n-label multiclass classification, though work in multiclass classification does not always effectively extend to this limit with full stratification. The essential difference between ranking and n-label multiclass classification is that ranking provides an ordering of elements as opposed to simply distinguishing them as in an n-label classification problem.

Research into the ranking problem for Machine Learning originated with the challenge of information retrieval[2,3], but has extended to general purpose ranking algorithms which apply a variety of established Machine Learning techniques to the unique challenges of ranking[4]. Considering ranking to solve the problem of learning preference, the many modern applications become clear. Some examples include ordering job applicants and providing more personalized suggestions for anything from restaurants to video games[5,6].

The ubiquity of data provides many opportunities to apply machine learning ranking techniques, but also raises a number of risks relating to unfair models leading to deleterious effects. With these algorithms becoming more established in everyday applications, it is important to remain cognizant of their limitations, as well as actively research methods to improve

performance outcomes for sensitive scenarios. Though this paper specifically focuses on pairwise boosting algorithms, many other strategies for ranking exist, including Support Vector Machines (SVM)[2], listwise boosting[7], graph-based algorithms[8], and neural networks[9].

1.1.1 Formal Ranking Nomenclature

The domain space of the ranking problem is the set X of items to be ranked, where each element x_i is referred to as an *instance*. The characteristic or quality that the ranking establishes is called the *ranking feature*[4]. For example, a search for nearby restaurants would consider a variety of characteristics to suggest a preferential ordering based on the available data, and the ranking feature can be abstracted as, "given the available information, which restaurant is the best choice?"

The goal of the ranking problem is to learn a model which uses a set of m element features, f_1 to f_m, to assign rankings to unlabeled sets of elements given their feature values[4]. Focusing on supervised learning, we train the model on a ranked set of elements which should have a representative distribution of the feature values which comprise the domain space. The model will ultimately use the feature values of the new data to determine how it should be ordered based on what patterns were compiled by the learning algorithm through the training process. Therefore, each feature can be viewed as an individual, primitive ranking function that contributes to the overall ranking of the set of elements.

We can describe each primitive ranking function as f_i where $f_i(x_a) > f_i(x_b)$ means that instance x_a is preferred over x_b by feature f_i[4]. Unlike the overall ranking function, primitive ranking functions can abstain from ranking particular instances; therefore $f_i(x_j) = \perp$ indicates that instance x_j was not given a ranking for feature f_i. Each primitive ranking function is summarized as $f_i : X$ with the additional element $\perp$. The overall ranking function can then be formalized as

$H : X$ without the additional element $\perp$ since all instances must have an overall ranking. For ranking function H, $H(x_a) > H(x_b)$ means that instance x_a is preferred over x_b.

1.1.2 Information Retrieval

Early applications of the ranking problem arose due to the nascent Internet and the need to return ordered responses to a query, which falls into the category of information retrieval. Google's PageRank algorithm can be thought of as an unsupervised learning algorithm which used backlinks between pages and could incorporate user preferences to recursively assign a value, then translated to a rank, to different sites in response to a query[3]. From there, ranking has developed into a full-fledged subfield of machine learning problems, but often retrains this initial association. Since the ranking problem is often framed as a corollary of the information retrieval problem, it is important to address the assumptions made in many information retrieval applications which do not necessarily hold for the general ranking problem.

In particular, information retrieval problems generally prioritize the accuracy of the most preferred elements, which can be referred to as "Top-K" accuracy[10]. Consider searching for a website out of millions of possibilities – the first page of results is what matters, and to a much lesser extent the second page of results, but how much does the 200^{th} page of possibilities actually matter? With the corresponding introduction of discounted accuracy and reward metrics, especially Normalized Discounted Cumulative Gain (NDCG), the ranking problem has been aligned in this paradigm which weights misranks based on their overall position. However, in the general ranking problem this is not necessarily the case, especially in pairwise algorithms. For example, consider a ranking of the best colleges in the United States; while the top slots connote the most prestige, many applicants will actually want to gauge the relative quality of schools further down the list. In this case, to provide useful information to users, a pairwise ranking algorithm might outperform a Top-K algorithm. Therefore, although Top-K ranking is

a very important application, it solves a specialized problem distinct from the cost-sensitive framework for ranking introduced in this paper.

1.1.3 Use of Boosting for Ranking

Boosting algorithms were an unexpected breakthrough in classification, with their seemingly miraculous empirical success being well-supported by extensive theoretical justification. The extension of boosting to ranking mostly leans on the classification theory, with a few seminal papers supporting the application of the technique to ranking. The iconic boosting algorithm for ranking is the appropriately named RankBoost[4], which has a discrete and a continuous variant. Its cousin, AdaRank, is more closely aligned by design with the classification boosting algorithm AdaBoost from which the inspiration for both is derived[7]. Boosting is a popular technique for ranking, but the learning problem introduces many complexities beyond the challenges for classification. Recent work has suggested that there are unaddressed differences in theory between boosting for classification and boosting for ranking that undermine the theoretical dominance of RankBoost. One of the primary criticisms is the uneven weighting of ties by RankBoost, which introduces a dimension of error not analogous to anything in the classification learning problem[11].

Boosting algorithms work by creating weighted ensemble classifiers[1]. Often, this is done with weak learners, such as decision stumps. Boosting theory asserts that classifiers with error less than chance can be combined using this technique into powerful classifiers with very complex decision surfaces. As opposed to bagging, another ensemble learning technique which using bootstrapping to assemble a quorum of voting classifiers, boosting works by reweighting the importance of the training data examples based on the results of the most recently added (weak) learner in the case of RankBoost, and the entire ensemble so far in the case of AdaRank[4,7]. For the ranking learning problem, training is usually done pairwise by testing if a classifier orders

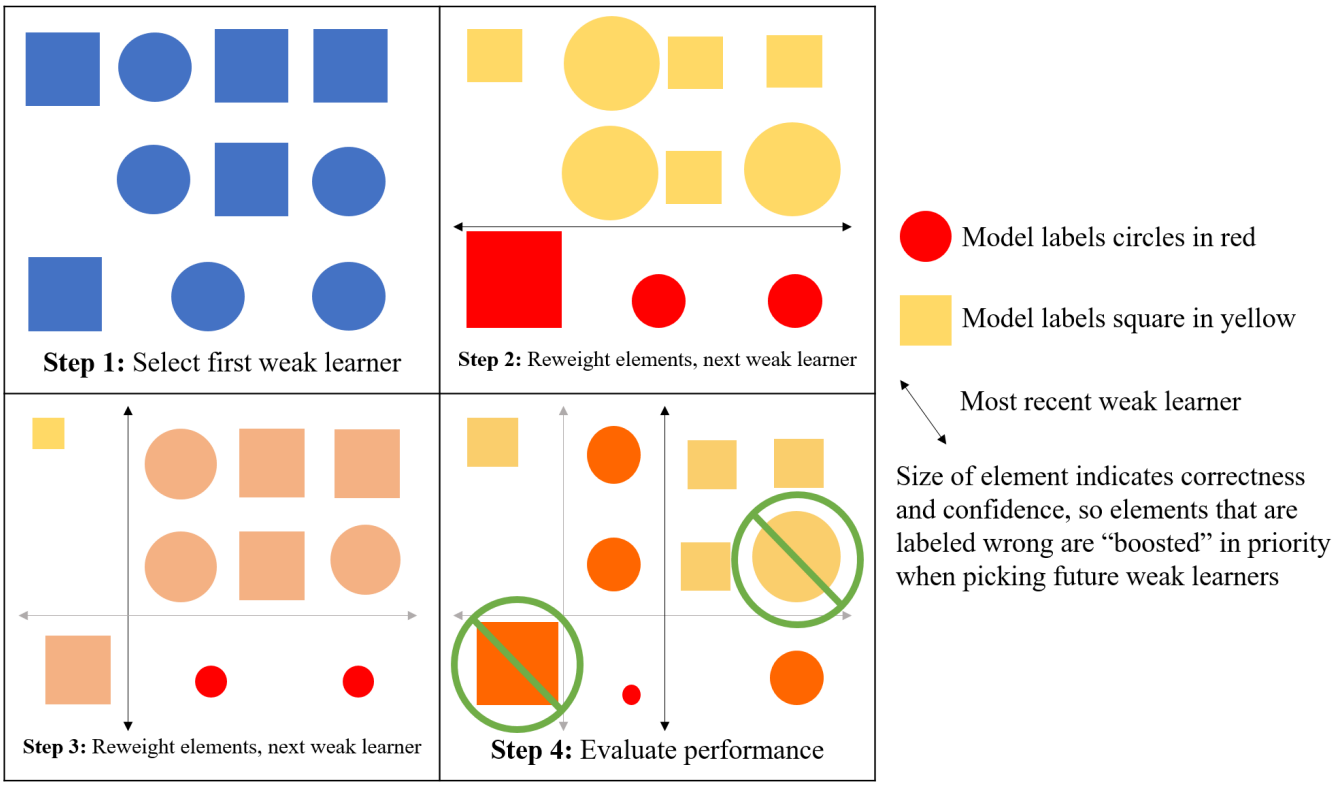

Figure 1.1. Visualization of Boosting for Classification

a pair of examples correctly. If the examples are ranked correctly by the newest learner to be incorporated into the ensemble, the pair's importance for the next iteration is decreased. Conversely, if a pair of examples is ranked incorrectly by this classifier, the weight of the pair will be increased. By this mechanism, boosting increases focus on pairs that are challenging to weight correctly and ensures that classifiers with diverse ranking priority are included in the ensemble.

The different in approach which distinguishes versions of boosting for ranking is how to reassign the weight of pairs which are tied by a given learner. Ties are especially prevalent for binary weak rankers, often used for ranking with boosting algorithms, which measure quality as either a 1 or a 0. Therefore, the method for adjusting the weights of training example pairs after adding a new classifier to the ensemble makes a significant difference in the resulting boosted classifier. RankBoost gives a variety of options for, and some empirical arguments to justify, its treatment

of ties but does not provide theoretical rationale to support the impact on training efficacy. The algorithm furthermore uses multiple error functions which treat ties in different ways, with the discrete version treating ties as errors and the continuous version giving ties a weight of 1. The discrete version is more closely tied to AdaBoost theory for classification[4], as it allows ties to be incorporated into the error, but empirical studies have shown that the continuous version of RankBoost outperforms the discrete version[11]. New research into how to reconcile the two has resulted in RankBoost+, which standardizes the treatment of ties to be the average reweighting applied to correct and incorrect pair outcomes[11].

Boosting is a flexible technique, and other research has combined it with existing ranking algorithms in order to enhance performance. Some examples, only partially based on the entertainment value of the names, are described below. McRank attempts to rank using classification boosting, with surprising success but atrocious time complexity[12]. The LambdaSMART algorithm combines boosting with the LambdaRank information retrieval training algorithm[9]. LambdaRank works by training a neural net on the gradient of the cost of the current model's ranking after the sort at each iteration. By combining this technique with the boosted regression tree modeling employed by MART, LambdaSMART achieves comparable accuracy to its progenitor algorithms with improved time complexity by tacking boosting onto a more powerful performance metric. However, as described above, these algorithms lack theoretical justification specific to ranking as preference learning cannot be perfectly converted to a classification problem, no matter how good their empirical results may be.

1.1.4 Differences from Ranking for Classification

Ranking for classification shares a number of properties with the ranking problem, especially for multiclass classification where elements must be stratified into more than two groups and therefore have stricter margin requirements[13]. However, ranking for classification is often used

for horizontal separation between groups that have no implicit ordering. Therefore, the ranked clusters which correspond to the class labels may not share the notions of quality and preference which are central to the ranking problem. Nonetheless, because of structural similarities between the problems and the availability of cost-sensitive datasets for multiclass classification, it is worth comparing work in cost-sensitive ranking algorithms to applications for multiclass classification[14–18]. A brief analysis on multiclass bias can be found in 2.3, and experimental results for cost-sensitive multiclass classification using ranking algorithms are described in 4.3.2.

1.2 Overview of Bias

Bias is an overloaded term for the abstract concept of some sort of inequity. Unfortunately, the definition of bias within ML is similarly fluid, which makes attempting to remove bias from ML algorithms or models an evolving challenge. Bias has been referred to by many different names in the literature, including disparate impact, indirect discrimination, redlining, statistical parity, disparate mistreatment, and equality of opportunity[19]. These terms have slightly different implications, and many include politically charged language which helps to draw necessary attention to this topic but may also serve as a distraction. In some situations, bias in the model may be glaring, but there is no guarantee, especially with more subtle models, that users will "know it when they see it." The more optimistic descriptors of this line of work are variants on Fairness for Machine Learning[20], but fairness is similarly nebulous and is generally defined as a lack of bias in its various forms anyway[21]. As with most machine learning problems, how the objective is framed can have a significant impact on the outcomes. As such, understanding the different causes of bias – and what bias even means in each context – is imperative in order to understand the implications of different bias-detection and mitigation techniques. Furthermore, each approach will necessarily have some tradeoff which will result in the standard approach likely beating the bias-reducing approach in certain performance metrics, e.g. accuracy

and recall[20]. Choosing the appropriate balance between fairness and model performance is an ongoing area of research which will often be specific to each scenario. However, finding performance bounds and evaluating the mathematical properties of different approaches can bolster the art of machine learning with a healthy dose of science.

1.2.1 Biased Datasets

Machine Learning works by recognizing and amplifying feature patterns in data in pursuit of some end, say, classification of previously unseen elements. The associated risk is that the models will encode problematic patterns that are systemic in the data, leading to results with embedded bias. Being cognizant of bias while training models, and performing model and dataset evaluation to detect this potential bias, is imperative in sensitive ML applications which could enforce existing or introduce new discriminatory structures with biased models. The Machine Learning community came under heavy public criticism in 2016 after Pro Publica revealed problematic trends in the COMPAS recidivism model, which according to some definitions of fairness learned patterns which disproportionately penalized black prisoners as compared to white prisoners eligible for parole[22,23]. The COMPAS team defended their model based on equality in a different error metric, highlighting how interpretation of results from different lenses can significantly alter perception about outcomes[24]. A few researchers had worked in fair ML before this negative publicity, but work in the research area sharply increased after this attention. Observing the citations on Dwork's seminal 2011 "Fairness through Awareness" paper[21], this change is exemplified. The paper was cited by 32 research papers in 2016, and 203 in 2019. This increase in interest draws vital focus to this area, but also risks fragmenting the consensus about appropriate methodology and drowning revolutionary improvements in flawed conventions. Understanding the foundations of this research is important, as is a fluency in the pros and cons of the most common approaches to bias mitigation currently in vogue.

1.2.2 Proposed Fairness Metrics

Many fairness metrics have been proposed for training ML classifiers, which are essentially ways of quantifying bias in the treatment of a certain group of elements. This bias must be detected or minimized depending on the context, and is done through pre-processing, modifications to the algorithm, or post-processing[20]. Pre-processing generally refers to the modification of the dataset to eliminate bias so that the training will not institute patterns of the bias from the data into the model. Algorithmic modifications change the process of training to either pick better models in terms of fairness to expand upon, like a genetic evolution process, or to actually train models with bias mitigated. Post-processing can be used to audit, and in some cases adjust, a model for fairness. These techniques can be combined and are not always clear-cut. For example, the dataset may be modified in a way which is compatible with changes to the algorithm, and then the resulting model could be audited and tweaked after training is complete. Friedler, et al. provide a comparative survey of several classification algorithms which raises concerns about preprocessing and data encoding, stability to dataset changes, and the proliferation of similar techniques[20]. However, regardless of which technique to detect and remove bias is used, the way bias is defined will fundamentally affect the outcome of the training. Many different options for quantifying bias exist, and more are being explored by current research being done in the field, but even a basic foundation of classification theory will provide vital background for the exploration of bias mitigation for ranking to come in a subsequent section.

There are several overlapping subcategories of bias metrics for training fair classification models. One of the most fundamental is blind versus aware training, also known as treatment versus impact parity[25]. In the blind training, the protected characteristic is completely expunged from the data, and these differences are not taken into account at all in training the model. The name

treatment parity arises because all elements are being treated in the same way. This can be useful when no implicit indicators or biases exist between the protected and non-protected class. For example, to fairly audition musicians for an orchestra a screen is used to make process blind so a candidate's race, gender, etc. cannot impact the decision, which should be based solely on the quality of their musical talent[26]. When discrimination is shallow and based on concealable external factors, this approach is valid. However, in many cases the opposite is true, and the model should be aware of the protected class in order to actively avoid deeper institutionalized disparity[27]. In these cases, the treatment of the protected class versus non-protected class is explicitly evaluated and adjusted to achieve the appropriate fairness technique, in order to equalize impact. In most cases where ML is being used for a sensitive application, this category will be more relevant based on extant patterns which could be learned as proxies for the protected class.

Another dimension of bias metrics is whether they are preference-based versus parity-based, as shown in Figure 1.2. Because parity-based is more popular, and has more variations, it is useful to first consider preference-based techniques. These strategies are based on Game Theory and Economics, and rely on the intuition that fairness can be introduced without necessarily striving for complete equity[19]. In Economics, competing parties can rarely attain exactly the same utility, but will nonetheless choose the best outcome for their constituency regardless of the effects on outside groups. Essentially, a group is incentivized to support an intervention as long as that group derives relative benefit from the change compared to its own alternative outcome, even if that intervention benefits another group more. Zafar, et al. define utility differently, where utility is the performance of the classifier $\mathbb{E}_{\mathbf{x},y}[\mathbb{I}(h(x) == y)]$ and group benefit is how shared attribute groups optimize their outcomes $\mathbb{E}_{\mathbf{x}|y}[\mathbb{I}(h(x) == 1)]$ where 1 is the preferable classification and y is the correct label for each element[19]. Here, $\mathbb{I}$ is an indicator function which returns if classifier h correctly labeled element x. The total utility of the models is the

aggregated accuracy across all groups.

According to Zafar, et al., preference-based fairness has several benefits, as it is more flexible to biased datasets and in general has a less severe impact on the accuracy or recall of a model's performance[19]. Essentially, different models are trained to accomplish the same task for the protected and non-protected groups. Each group is judged or classified according to the preferred model for that group. In the limit, this could be extended to increasingly precise subgroups or even to individuals, though this would quickly become prohibitively computationally expensive. This technique is also known as group envy-freeness, which means that no group would collectively benefit by evaluation using a different classifier. Therefore, the protected class is being treated, if not equitably, better than would otherwise have been the case if its elements were being evaluated by a different feasible model. The models for each group can be selected using linear programming using a disciplined convex-concave program (DCCP) approximation, meaning this non-convex problem is represented as the sum of a convex and a concave term with a general solution[19].

$$\underset{h_z}{\text{minimize}} \qquad \frac{1}{N}\sum_{(\mathbf{x},y,z)} l_{h_z}(\mathbf{x},y) + \sum_{z\in\mathcal{Z}} \lambda_z \Omega(h_z)$$

$$\text{subject to} \qquad \sum_{\mathbf{x}\in\mathcal{D}_z} \max\left(0, h_z^T\mathbf{x}\right) \geq \sum_{\mathbf{x}\in\mathcal{D}_z} \max\left(0, h'^T\mathbf{x}\right) \text{ for all } z\in\mathcal{Z}$$

Here, the loss $l_{h_z}(\mathbf{x},y)$ is a convex function under the constraint that the chosen classifier h_z for each class $z\in\mathcal{Z}$ outperforms the impact parity classifier h', where the classifiers are linear functions in the feature space. $\Omega(h_z)$ is a convex-regularizer to make this problem consistent with the requirements for DCCP approximation.

Parity-based bias metrics are more stringent about eliminating their respective unfair aspects, and therefore may have a bigger tradeoff with the general performance metrics of the model.

There are many options available, but this background overview will only cover the most well-known and relevant ones. In general, parity-based metrics advance more intuitive approaches to equality, but the specifics of each approach can still lead to wildly different outcomes. Recall the controversy over the COMPAS model of recidivism which was criticized for perceived inequality between its treatment of black and white convicts[22]. The researchers defended it by saying it was fair by *predictive parity*, which means that a label should result in the same true positive rate and false positive rate between groups, i.e. the probability of the label being correct should be the same for the protected class as everyone else[28]. However, false positive rates may differ due to class asymmetry, as in the case of the COMPAS model which overrepresented black prisoners and therefore was much less likely to offer parole when appropriate than it would be to parole a comparable white prisoner[24]. This example, and the justified resulting controversy, should highlight the potential problems with using an intuitive metric inappropriate to the circumstance or dataset available.

Next, consider *conditional independence*, also called equalized odds[29]. Conditional independence means that conditioned on the correct label and the protected class, the expected model output should be the same as when it is just conditioned on the correct label.

$$E[h(x)|A = a, Y = y] = E[h(x)|Y = y]$$

Therefore, all elements x in each classification are expected to receive the same treatment from the classifier h given their status as members of the protected class a and their correct label y, where A is the set of protected and non-protected class as determined by some protected characteristic and Y is the set of classifications. However, similarly to predictive parity, controversy over outcomes may arise if the protected class is not distributed between classifications in the same way as the other elements are, since differences in the performance of the model

for different labels could then disproportionately affect one class. Observe the following toy example. There are 12 people who like spicy food, 8 of whom are women and 4 of whom are men, and 12 people who like sweet food, 8 of whom are men and 4 of whom are women. Consider a model that correctly labels someone who likes spicy food 75% of the time, and someone who likes sweet food 50% of the time, without difference based on gender within each label. 4 men who like sweet food will get their preference, and 1 man who likes spicy food will get sweet food. 2 women who like sweet food will get their preference, and 2 women who like spicy food will get sweet food. Of the 7 men who get spicy food, 4 will have wanted sweet food, but of the 8 women who get spicy food, only 2 would have preferred sweet food. 5 men will be denied their preference, versus 4 women. If spicy food is considered a positive label, the FPR for women is 25%, versus about 57% for men. Conversely, the FNR is 50% for women and 20% for men. Although the stakes are low in this toy example, real life applications can imply much more serious consequences for these treatment differences in the face of asymmetric data, which will not be detected by conditional independence.

Another popular and intuitive parity-based bias metric is *statistical independence*, which can also be called group fairness or demographic parity[29]. It states that statistically, having the protected characteristic should have no impact on the treatment of an element as compared to the rest of the elements in the dataset. That is to say, the protected class gets the same treatment as the average element according to the equation

$$E[h(X)|A = a] = E[h(X)]$$

where h is the classifier and $A = a$ denotes membership in the protected class. However, as with equalized odds, there are a number of potent criticisms of using statistical independence to determine fairness. Consider the following three major pitfalls described by Dwork, et al. in "Fairness through Awareness"[21]. One, there may be suboptimal equilibria selected by this

fairness metric in which all groups are mistreated equally. Two, self-fulfilling prophecy is where elements of the protected class are treated in ways that set them up to fail rather than being classified fairly by the model. Three, subgroup targeting, which is analogous to gerrymandering of subgroups, means the system can be rigged while still appearing fair according to this metric[30]. These criticisms highlight the shortcomings of using group fairness metrics which may eclipse the qualities and appropriate sorting of individual elements, even to the extent that the efforts to increase fairness end up being universally counterproductive.

Calibration is another fairness metric which has been explored for classification. It uses the confidence of the model to evaluate whether the sorting is effective and reflective of the quality of the decision. In this case, the probability of correctness should actually match the probability or confidence assigned by the model[31]. Therefore, if the model expresses 75% confidence about 400 elements, 300 of them should be classified correctly. Calibration can draw attention to models which are less well-suited to treatment of the protected class if the confidence or calibration quality significantly differ between elements with the protected characteristic and those without it. However, it is more difficult to use calibration for training and correction, so its primary utility is as a tool for auditing the processes of a potentially biased model by essentially breaking into the black box.

Several of the problems introduced for the group fairness metrics explored above can be mitigated to some extent by using subgroup fairness metrics. These metrics extend the notion of fairness from the entirety of the protected class to smaller subsets which can improve outcomes. In the limit, infinitely precise subgroups are computationally intractable, but research has shown that performance improves significantly with even a few layers of subgroup analysis[32]. Intersectionality has also been a focus of some research efforts. By using subgroups it is possible to reduce disparate impacts between members of each protected class, particularly

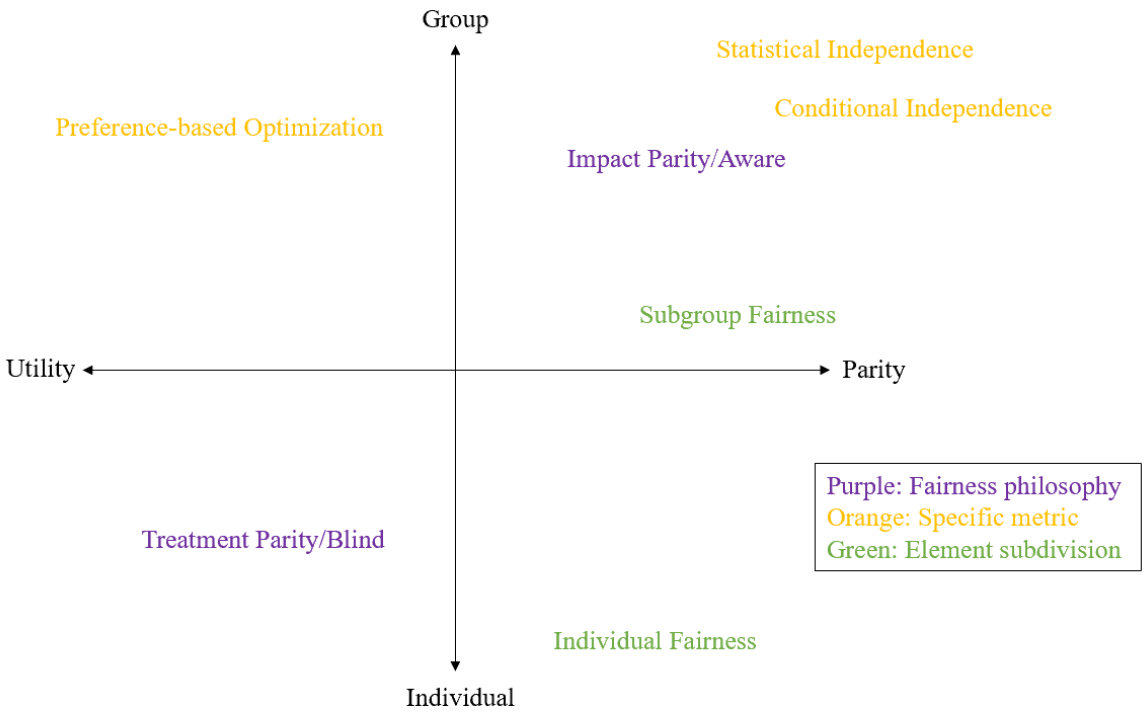

Figure 1.2. Properties of Basic Fairness Metrics

when there is overlap between multiple protected characteristics[32]. For example, consider a set of colored shapes, equally distributed between circles and squares. Each shape comes in blue or green, with exactly half of each color. If square and blue are the protected classes, without considering subgroup fairness or intersectionality, selecting all the green squares and all the blue circles would be considered fair by group fairness metrics[33]. However, this would not equally represent the four different subcategories considered when both shape and color are taken into account. The number of different subgroups in consideration can grow combinatorically, but research into classification subgroup fairness heuristics has shown reasonable success in identifying tractable approaches.

Rather than computing fairness across subgroups, in some cases it may be more appropriate

to guarantee individual fairness. In 2011 Dwork et. al proposed that "similar individuals should be treated similarly."[21] This does leave ambiguity to define what constitutes similarity of elements and treatment, especially when a protected characteristic is involved and needs to be taken into consideration. In that original paper, a new fairness metric must be devised for each classification task, making it difficult to generalize. Additionally, even with an appropriate metric at hand, individual fairness is difficult to compute at the limit. Overcoming these challenges seems worthwhile for some sensitive applications where using group or subgroup fairness criteria may lead to problematic externalities and conceal injustice within the treatment of a given group or subgroup. One proposal is to pursue average individual fairness, in which individuals undergo a series of similar classifications, which as a whole do not systematically disadvantage them[33]. Therefore, even though each specific classification task may not satisfy the requirements for fairness of all individuals, in the limit of sufficient classification tasks applied, everyone has on average been treated equitably according to their individual qualifications, making bias into random rather than systematic unfairness. Refer to Figure 1.2 to see where various fairness metrics fall on this spectrum between group and individual fairness.

There is a vast corpus of recent research into fairness for ML classification, and providing a comprehensive literature review is becoming increasingly infeasible. However, the broad categorizations and explorations of the different dimensions of a fairness metric should provide adequate insight to approach the problem of bias mitigation for ranking. There are a few papers currently published on this topic, which will be introduced formally in Chapter2, all of which rely heavily on the avenues of defining bias for classification explained above. For the most part, the metrics for classification go unchallenged in this handful of papers, so the following section presents a number of of the shortcomings which call into question whether these classification metrics and techniques should be applied to fair ranking attempts.

1.2.3 Reductions Approach to Training Fair Classifiers

Agarwal, et al. propose a method to train a fair classifier applicable to a variety of bias metrics using cost-sensitive intermediate classifiers[29]. The bias criteria must be expressible as a set of linear constraints, and the problem is solved by selecting the classifier with the lowest empirical error that satisfies the fairness requirement.

$$\min_{Q \text{ exist on } \Delta} \widehat{err}(Q) \qquad \text{subject to} \qquad M\hat{\mu}(Q) \leq \hat{c}$$

Here, using the training data the goal is to minimize the empirical error of the randomized classifier Q from Δ, the set of all distributions of classifiers. The real matrix M multiplies the vector of moments $\hat{\mu}$ defined by the classifier on each element, subject to the linear constraint vector $\hat{c}$. The data is assigned costs and treated as a weighted classification problem, for which several efficient classification algorithms exist. The authors reconstruct their constraint equation as a Langragian and solve the ν-approximate saddlepoint problem (where ν is a precision hyperparameter selected by the user) to pick the best classifier which satisfies the given fairness constraints. The cost-sensitive classification algorithm is used to compute the best response function to minimize $L(Q, \lambda)$. Here, L is the Lagrangian function, λ is the vector of Lagrange multipliers corresponding to the fairness constraints imposed upon the problem, and Q is the randomized classifier being evaluated. This connection between cost-sensitivity and bias constructed as a linear program to select the best classifier suggests renewed utility for research into asymmetric machine learning. Furthermore, the potential use of cost-sensitive algorithm variants to optimize fair results drives the focus on cost-sensitive ranking described in Chapter 3. Ultimately, using cost-sensitive machine learning is a reasonable approach to fair machine learning because both pursue the mitigation of risk, which implies that a similar transformation of ranking bias to cost could provide a more robust solution for fair ranking than extending group statistical metrics from classification.

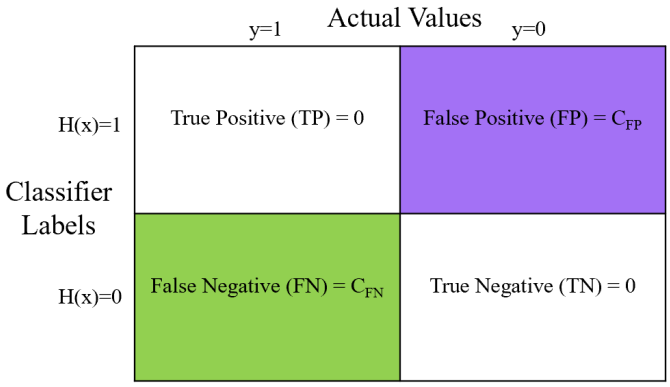

Figure 1.3. Asymmetric Confusion Matrix

1.3 Asymmetric Machine Learning

Machine learning captures patterns in the data, but datasets are rarely equally distributed. Cost- and class-imbalance may impact the desired model structure, which needs to be built into the training of the model[1]. Many algorithms that do not specifically account for dataset asymmetry may train models favoring the dominant class in the dataset. Erring on the side of probability by selecting the more common label for a given class for elements on the margin weakens the model, which is supposed to be capturing the distinguishing features, not guessing the more common outcome[17]. In cases with cost asymmetry, where false positives and false negatives may have significantly different consequences, it is similarly important to focus the model according to the cost structure[22]. For example, with medical testing, a false negative is often much more severe than a false positive, which can be resolved with further evaluation. Cost can dictate how the model should behave when faced with edge cases, and the degree of caution with which close calls should be treated.

The computation to incorporate cost and the one to deal with class asymmetry differ, but they can be transformed in either direction to the other with relative ease[1]. This interchangeability

can be extended to multiclass problems as well. Therefore, mathematically cost- and class-asymmetry can be considered equivalent and proofs about one can be trivially extended to demonstrate properties of both. This interchangeability also implies that cost can be applied to compensate for class asymmetry when training a model. Because class asymmetry is often a central cause of bias in a model, its consideration during training may be integral to preventing biased treatment of the less populous class. The use of cost to strengthen the focus of the model being trained on the class asymmetry may lead to better definition along the margin. Additionally, based on the findings of Agarwal, et al. described above, this work may be extensible to bias mitigation and fairness problems[29].

1.3.1 Rank Agnostic Cost

In some cases, the quality of the ordering at the bottom of a ranking of elements is much less important than the quality at the top of the ranking. For example, when querying millions of websites in a search engine, only the results on the first page or two really matter. Page 150 and page 1500 could be swapped without any impact on the user because those results are not being used. As a result, a "Top-K" approach is taken in many ranking applications[6]. The associated metrics disregard any results that come after a threshold, the first K results. Discounting quality metrics, especially NDCG, also reflect this paradigm by weighting the correctness of the highest ranked elements much more than the correctness of the last elements[10]. However, this means that the treatment of the elements that are rated near the bottom is much more haphazard, especially when metrics such as NDCG are actually used to train the model. LambdaRank actually uses the cost-gradient from NDCG to develop the cost-aware ranking model, but this heavily focuses on only the top elements due to the rapid discounting of importance as rank index increases[9]. While appropriate in certain circumstances, in other situations a position-agnostic method of specifying importance is required.

Some rankings require precision of placement for certain elements which do not all fall at the top of the list. Top-K approaches and discounted efforts actively neglect the bottom of the ordering, while unweighted strategies may fail to prioritize these pivotal elements. When a subset of elements is prioritized for correct ranking which does not coincide with just the highest ranked elements in the list, a different cost structure is useful. This approach can be thought of as "rank agnostic cost" since it does not depend on the position of an element in the ordering when determining its importance. Having this cost-structure can be useful in many scenarios, including potentially in bias-reduction for ranking. The cost will focus the training of the model on specific pairs or individual elements which will have to be distinguished. Similar to a support-vector machine, this approach will force the training process to correctly rank very important elements or to focus on correctly ranking elements along the margin.

2 Bias Mitigation for Ranking

Although the machine learning literature is dominated by discussion of classification, with bias and fairness theory following the same trend, there has been recent interest in achieving fair ranking models. Though more challenging to quantify and train than fair classification models, bias-mitigated rankers are imperative based on the widespread proliferation of sensitive ML ranking applications. With an unprecedented availability of data, there is incredible opportunity to create more efficient and equitable systems, but only if the models themselves do not incorporate systematic biases. Several papers have been published within the last year related to this topic, meriting an overview in this section. These papers do explore some interesting points and approaches, but in general the proposals over-rely on classification theory and lack mathematical justification for intuitive but inadequate fairness metrics. This chapter proposes a framework to evaluate metrics for fair ranking in 2.2.1 and demonstrates how each of the existing methods in use for fair ranking falls short according to this framework. Specifically, this section of the paper investigates the pitfalls of statistical independence (2.2.2), conditional independence (2.2.3), calibration (2.2.4), and preference-based fairness (2.2.5), for fair ranking problems. Additionally, 2.2.6 considers subgroup and individual fairness for ranking, and 2.3 goes into the similarities and differences with reducing bias for multiclass classification. Although this paper critiques the methodology of current work in this area, those contributions to this fledgling research area are an important first step in getting ranking fairness the research focus it deserves.

Several papers provide tools for bias mitigation which require the user to define a fairness metric which the described algorithm will then optimize. The Mithra webtool calls this the "goodness criterion," then provides a framework which uses that criterion to rank data[34]. Asudeh, et al. take a similar approach, determining the closest satisfying function given the user-provided definition of bias[35]. Providing post-processing algorithms to optimize some fairness criterion is important, but insufficient while the research on appropriate bias metrics is so underdeveloped. Other papers employ classification metrics without modification for ranking. Castillo gives a concise overview of fairness for information retrieval which incorporates several useful ideas for that problem space, but relies on "sufficient presence" (statistical parity), "consistent treatment" (equalized odds), and "proper representation" (dataset quality)[6]. The suggestions for fairness metrics in information retrieval, namely attention-based and probability-based, are interesting, but outside the scope of this paper which is focused on the general ranking problem rather than information retrieval. Similarly, the work by Zehlike, et al. resembles the information retrieval problem as they create a top-K fair ranking algorithm, where the ranking of the top elements should reflect quality but also meet a minimum proportion of the protected class and produce a consistent ordering within the top-K elements[36]. Their approach, dubbed FA*IR, is useful in many scenarios since it has relatively low detriment to the utility of the top-K selection versus ranking algorithms without fairness intervention, but more limited than the general fair ranking problem explored here. LinkedIn has also been considering the fairness of its ranking algorithms, but the paper focuses on describing a greedy search technique that aims to achieve basic classification parity metrics without considering the nuances of ranking[5].

Most relevant to this work is FARE, which provided ranking modifications of three of the basic classification metrics[31]. Though the work to prove the properties of these metrics was lacking in the paper, the authors' attention to the differences between ranking and classification

is commendable. They introduce a pairwise evaluation scheme which includes rank equality (analogous to statistical parity), rank calibration (considering inverted pairs containing an element of the protected class), and rank parity (similar to conditional parity, ideally around $\frac{1}{2}$). This approach allows auditing of a ranker, but does not incorporate these metrics into the training process. Additionally, it assumes an unbiased labeled dataset for the first two metrics, which are based on inversions of pairs according to the expected outcome. The technique also requires a well-distributed, unbiased dataset to accomplish rank parity, which is vulnerable to tokenization, clustering, and randomized treatment of the protected class (see 2.2.4). While considering the inadequacies of classification theory for various ranking scenarios, FARE's new definitions will run into many of the same challenges which are demonstrated in the following section.

2.1 Need for Fair Ranking

The exact metric for fairness in ranking may not be self-evident, but the obvious applications are myriad. Machine learning techniques are useful in many applications which require attention to sensitive considerations. In fact, use of algorithms to replace arbitrary processes and automate simple, tedious tasks can improve performance outcomes and allow humans to focus on more complicated tasks. The problem of fairness arises when the task at hand touches on some protected characteristic of some of the elements being ranked, which are called members of the protected class. Many natural examples will touch on politically-charged topics, particularly discrimination based on race or gender, but it is important to remember that the need for fairness in machine learning extends beyond social issues and the controversy they may engender. For example, information retrieval tasks may incorporate a very different notion of fairness which should still be compatible with any conclusions reached about the more intuitive social implications of using ML for ranking.

The need for fairness is exemplified in the job application process, where race, gender, criminal record, and status as a parent may all become relevant protected characteristics[37]. The issue of affirmative action is related, and connects to higher education as well. In order to maintain focus on the breadth of this issue, rather than the social issues that may come into play, for the purposes of a recurring example of fairness consider a slightly different college admission scenario. Disregard all other potential protected class groupings and concentrate on whether an applicant attended a public high school or a private high school. While this characteristic of an applicant is correlated with other factors, such as socioeconomic status and geographical location, for the purpose of conceptualization it is a sufficiently neutral example which may help to elucidate some of the nuance in determining a fairness metric.

2.1.1 Tokenization

Many challenges for fair ranking exist, but *tokenization*, where a subset of the protected class is systematically misranked in an extreme way, is both particularly difficult to overcome and problematic in its impact. Before delving into potential bias metrics which can be minimized, highlighting tokenization can introduce the potential pitfalls for a flawed metric. Avoiding tokenization is related to subgroup fairness for ranking, since members of the protected class need to be ranked appropriately by the trained model. Simply reducing some bias construct in no way guarantees satisfactory outcomes, especially if the algorithm just trains a differently biased ranker which optimizes the fairness metric in some counterproductive pattern. For example, consider the problems with weighting a few "token" elements very highly while discriminating against others of the protected class. A couple of elements may be lifted up in rank to balance out the biased pattern, but the underlying structure of bias could easily prevail in this scenario, which can be connected to the "model minority" problem and the "glass ceiling." Furthermore, even within this potentially fraught tokenization system, there is a clear need to avoid treating

members of the protected class randomly. Minimizing the bias metric should not significantly reduce the efficacy of the preference ordering for elements with protected characteristics.

2.2 Shortcomings of Classification Theory

Because ranking separates each element into a separate preference level with respect to all other elements, the relationship between elements of different broad groups becomes much more complex. In classification applications, unfairness in the model can be obfuscated due to the prioritization of impact (i.e. what label the model gives to each element is more important than how the model determines that label). The effect of stratification due to ranking is that there is no longer a grouping of elements under a label which mitigates impact unfairness in the model. To illustrate this, consider a pass-fail grading scheme versus a numerical score out of 100. The introduction of higher precision feedback means that, for example, a model that gives all passing elements of the protected class scores between 60-70 versus passing non-protected class elements all receiving 90-100 could achieve impact parity for binary classification but not for ranking. In other words, for classification there is more leeway for model bias that does not result in disparate impact because only elements that are not clearly in one category are subject to this discrimination for a high-performing model. Even if the model systematically under- or overrates elements of the protected class, as long as they are still classified correctly this will not be detected as bias. However, with ranking every element is substantially distinguished, so these hidden biases that can exist in classification are much more problematic for a ranker. Mathematically, this means that the use of broad group statistics common for bias mitigation in classification is inadequate for ranking due to the added nuance resulting from the goal of providing a complete ordering of all elements. In summary, the existing classification bias metrics for group fairness and current customizations for ranking fail to detect certain behavioral differences between rankings that cannot be considered equally fair (see Observations 1, 2, and 3). In the following sections, the weaknesses of a number of approaches are explained in more detail,

with ranking scenarios and characteristics that clearly demonstrate the need for improved bias metrics. Statistical and conditional independence are specifically chosen due to their usage in the limited fair ranking literature[5,31,36,38], which demonstrates the urgent need to challenge the shortcomings of these metrics before they become normative for fair ranking.

2.2.1 Terminology to Evaluate Unfairness in Ranking

Due to the limited research in this area and the lack of extant theory for evaluating the quality of a ranking fairness metric, before going through the prominent candidates it is important to develop a common vocabulary to describe performance. Without having a fixed candidate for fair ranking, rather than capturing positive aspects of the ranking fairness possibilities these heuristics will describe potential failures of fairness. Considering the best, worst, and average cases will demonstrate the extent to which these metrics satisfy these necessary properties, or fail to do so. Because of the complete stratification of elements into a preferential ordering, many of the biases which can be hidden or disregarded in a binary classification setting come into play. Depending on the context, these may not prevent these existing classification-turned-ranking fairness metrics from being appropriate. However, it is important to be aware of how they break down in order to avoid misuse of flawed bias metrics in sensitive ranking situations.

For the purposes of evaluation, these heuristics will be defined based on the codification of relatively intuitive notions of fairness, i.e. what would apply in a societal context. They are not intended to be a universal, sufficient framework for what is considered fair or unfair, but rather a starting place from which to understand the limitations of existing bias metrics. With those caveats, the following terms will be applied to the assessment of several bias metrics based on their immediate applicability and current usage for the problem of fair ranking. Note that detection of a property applies if scenarios with significantly different behavior related to that

property result in different scores by a given bias metric.

Property 1: A smooth (uniform) distribution of protected class elements should be detectable. Formally, for otherwise equivalent rankers according to the bias metric, the metric with the more uniform distribution should be rated as less biased.

While a smooth distribution of protected elements may not always be context-appropriate, a generic fairness metric for ranking should be able to detect and optimize for this property. If the protected characteristic plays no role in the ordering of the elements, the protected class can be expected to be randomly distributed throughout the ranking, which in the average case will manifest as an approximately even spread. For any position in the final ranking, the expectation that it be a member of the protected class should be equal to the proportion of the population included in the protected class. Consider again the example about assigning numerical grades to students versus pass/fail binary classification. If, for a sufficiently large sample size, no students from the specified category (which should not correlate with ranking performance) receive a grade within a certain range, this might raise a red flag. More concretely, in a large lecture class of hundreds of students with an 80/20 male/female ratio where the majority of students get scores between 70-90, if the ranker predicts that no female students received a grade between 75-85, this pattern might raise concerns even if some bias metric indicates that overall female students are not underperforming compared to male students. Here, $H(x)$ is the final rank an individual element x receives from the ensemble ranker, i is each possible rank for the set of elements X, A refers to the set of elements in the protected class, N is the total number of elements being ranked, and $x_A = 1$ if x is a member of the protected class A.

$$\mathbb{E}[H(x) = i, x_A = 1] = \frac{|\mathbf{x}_A|}{N}$$

Definition 1: Tokenization occurs when the exaggerated high ranking of some elements of the protected class cancels out the exaggerated low ranking of the majority of such elements.

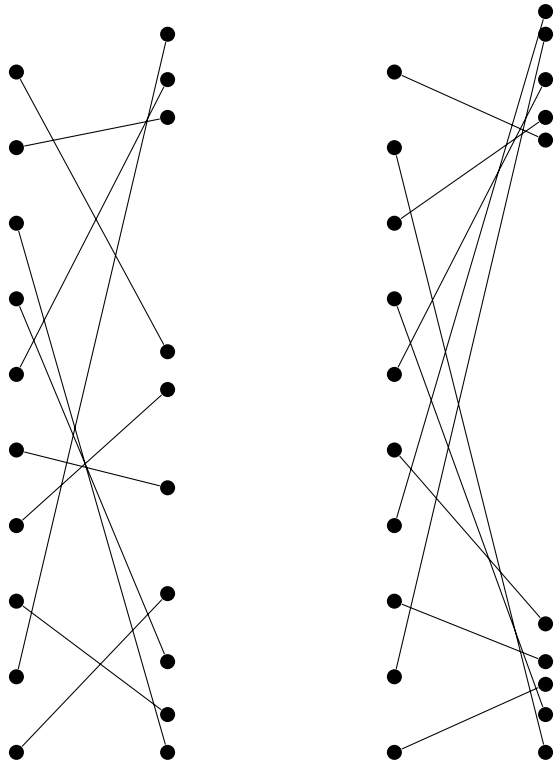

Figure 2.1. Visualizations of Tokenization and Bimodal Ranking

Note that exaggerated misranking refers to systematic incorrect ranking which is disproportionate to the general performance of the model across all elements being ranked.
Group fairness rarely refers to the potentially excessive elevation of some member(s) at the expense of all other members of that group. Therefore, bias metrics for ranking should be able to detect this type of pattern where some elements are "chosen" to satisfy the fairness metric to the detriment of other elements in the protected class.

Definition 2: Bimodal rankings are a special case of tokenization where elements of the protected class are ranked either very highly or very lowly.
A bias metric should be able to detect this type of behavior which in many contexts will indicate inconsistent treatment of the protected class. Bimodal rankings of the protected class may appear due to training a model that is too rigid in its ranking of the protected class, leading to

an artificial dichotomy.

Definition 3: Noisy ranking refers to a model which disproportionately incorrectly ranks elements of the protected class compared to elements lacking the protected characteristic.
This problem is especially likely in problems with high class asymmetry. The performance metric, i.e. some type of rank loss, may be optimized on the non-protected class elements, while the bias metric is optimized on the protected class elements. If the ranker can systematically misrank elements of the protected class without negatively impacting the bias metric, this will be referred to as permitting noisy ranking.

Definition 4: Arbitrary treatment is a type of error which occurs within the protected class where elements are treated as interchangable.
This is similar to both the issues of tokenization and noisy ranking discussed above, but specifically refers to failing to distinguish protected class elements from each other based on their respective qualities, leading to unfairness within the protected class.

Proposition 1. Sensitivity to class asymmetry and dataset bias is a threat to the generalization of certain bias metrics.
Recall the COMPAS dataset and the claim of predictive parity. Some fairness applications can rely on having balanced datasets, or achieve this prerequisite through preprocessing. Many real-world datasets cannot be assumed to avoid systemic biases, so class asymmetry is a reality which impacts ranking fairness problems.

Proposition 2. If the objective is treatment parity, where membership in the protected class does not impact the final ranking of an element, non-correlation between rank and protected class, i.e. arbitrary distribution of protected elements, is desirable. However, this distribution

cannot consistently satisfy the goal of impact parity for an individual ranker, where the protected class receives the same treatment according to some standard as the population at large, since many such non-correlated distributions will fail according to group fairness metrics. Statistical tests such as the runs test of randomness can be used to determine if distributions match within a certain degree of probability with respect to mean and variance[39]. If unbiased algorithms are trained on fair datasets then treatment parity could be sufficient; however, due to real-world concerns about the shortcomings of both algorithms and training data, impact parity is the priority for practical ML ranking problems. Detecting a random distribution of protected elements can show that the protected class is broadly not impacting its individual elements compared to their peers, but based on the logic described in Proposition 1 true randomness is not desired for most sensitive applications which require some degree of monitoring or intervention to ensure a balanced model with respect to impact on the protected class. Consider legal restrictions such as the federal 80% rule (four/fifths rule) for hiring, which determines that no minority group can be represented too disproportionately compared to any other group[40]. The cultural understanding of group fairness in this way contradicts ideas of individual fairness (i.e. treatment parity) because group representation, which cannot be guaranteed by random processes, is key.

The vocabulary introduced in this section will be used in the subsequent analyses of several potential bias metrics for ranking due to their applicability to the problem or reference in the existing literature explored above. More work should be done to create a comprehensive suite of heuristics which can apply to the many sensitive contexts where the fair ranking problem is applicable. However, for the purposes of an initial survey of attempts at fair ranking, these terms are sufficient to raise a number of potent concerns and hopefully motivate the development of more robust fairness metrics.

2.2.2 Statistical Independence

Extending statistical independence from classification to ranking evaluates whether membership in the protected class impacts the expected rank of an element[5,36].

$$\mathbb{E}[H(x)|A(x)] = \mathbb{E}[H(X)]$$

Essentially, this means the average rank should be the same for members of the protected class and members of the non-protected class, which would result in each class having an average rank of $\frac{n}{2}$ where n is the number of elements being ranked. While this approach can arguably provide important (albeit incomplete) insights for classification, since it means the separation of elements does not reflect their status as members of the protected class, use of expected values means that this metric cannot capture the specific importance of ordering in the ranking problem. There are many different distributions which share the same mean, so to guarantee that elements of each class have as their mean the midway point of the rank gives very little meaningful information.

Observation 1. Statistical independence does not detect a smooth distribution, tokenization, or arbitrary treatment of the protected class for ranking.

Proof: Consider a ranking of 100 elements, where 10 elements are members of the protected class. To guarantee statistical independence of their final rank, the average of their ranks should be 50. The following ranking results for the elements of the protected class all exactly fulfill this requirement.

- Mean: $45, 46, 47, 48, 49, 51, 52, 53, 54, 55$
- Bimodal: $1, 2, 3, 4, 5, 95, 96, 97, 98, 99$
- Bimodal: $2, 3, 5, 67, 68, 69, 70, 71, 72, 73$
- Uniform: $5, 15, 25, 35, 45, 55, 65, 75, 85, 95$

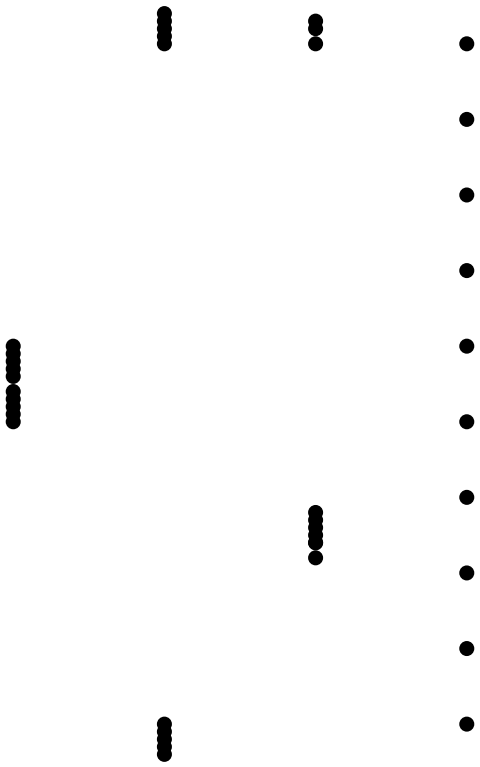

Figure 2.2. Four Rankings of Protected Class with Same Statistical Independence Score

Forcing a model to optimize for statistical independence can push it to these local "statistical bias minima" which are equivalent from the perspective of statistical independence. Depending on the context of the scenario, each of these may perpetuate systematic unfairness instead. For reflection on the fundamental challenges for use of statistical independence which also apply to classification, refer to 1.2.2, i.e. for the discussion of the potentially arbitrary treatment within the protected class.

2.2.3 Conditional Independence

Conditional independence for ranking can either be interpreted as the expected rank of an element being the same when given the true rank and the protected characteristic as when given just the true rank,

$$\mathbb{E}[H(x)|R(X), A(x)] = \mathbb{E}[H(x)|R(x)]$$

or as the pairwise classification by the weak rankers being conditionally independent of the protected characteristic[5],

$$\mathbb{E}[h(x)|y(x), A(x)] = \mathbb{E}[h(x)|y(x)]$$

Each of these approaches raises concerns about the ramifications for the fairness of the final ranking. The former possibility necessitates considering a distribution or error bar around the true rank which should not systematically overrank or underrank members of the protected class. The second approach of guaranteeing conditional independence on the weak rankers exerts very little influence over the final fairness of the ranking since the metric will not be coordinated between weak rankers and ties are not considered. Similar to statistical independence, above, this can lead to clustering or bimodal rankings due to tokenization or arbitrary treatment.

A variant of this approach, used in the research effort to make LinkedIn more fair, is to change the conditional independence so that it works as a "top-K" sort of approach where the equalized opportunity (another term for conditional independence) metric only evaluates the results that come up first[5]. This information retrieval approach somewhat matches the problem space since recruiters and LinkedIn users are probably only looking at the top few search results. However, the objections below still apply when the search space is shrunk to just those ranked at the top, since that can be considered a ranking subproblem. Without ensuring some degree of fine-grained fairness beyond just receiving a high rank, conditional independence will still have many shortcomings regarding ordering, error, and tokenization as described below. Notably, this model of group fairness also fails to account for intersectionality and subgroup fairness.

Observation 2. Conditional independence cannot detect tokenization and noisy ranking.

Proof: Consider the elements of the protected class described in the proof of Observation 1. For the first approach, determining conditional independence on the final ranking, observe three

critical scenarios which would satisfy this metric but confound the intent of ensuring fairness. Assume for this purpose that the fair rank is the uniform distribution. In Figure 2.3 these three scenarios are depicted side-by-side with the true (uniform) rank of the protected class on the left of each bipartite graph, one for each scenario, and the undesirable rank from some ranker on the right. The movement of the elements is shown using an edge, which provides the visual representation of the counterexamples described in the following propositions.

Proposition 3. Conditional Independence will not detect clustering, i.e. fail to distinguish between elements of the protected class in a "balanced" way, such that the negative impact is "canceled" by the positive impact, a type of systematic misranking of the protected class. Refer to column 1 of Figure 2.3.

$5 \rightarrow 14, 15 \rightarrow 15, 25 \rightarrow 16, 35 \rightarrow 44, 45 \rightarrow 45, 55 \rightarrow 46, 65 \rightarrow 79, 75 \rightarrow 80, 85 \rightarrow 81, 95 \rightarrow 82$

Proposition 4. Conditional Independence cannot detect noisy ranking, i.e. arbitrary ranking of the class with fewer elements. Refer to column 2 of Figure 2.3.

$5 \rightarrow 15, 15 \rightarrow 4, 25 \rightarrow 33, 35 \rightarrow 22, 45 \rightarrow 54, 55 \rightarrow 70, 65 \rightarrow 49, 75 \rightarrow 81, 85 \rightarrow 90, 95 \rightarrow 82$

Proposition 5. Conditional Independence cannot detect tokenization, i.e. systematically favoring the higher rated elements of the protected class and discriminating against the lower rated ones, another type of systematic misranking of the protected class. Refer to column 3 of Figure 2.3.

$5 \rightarrow 1, 15 \rightarrow 8, 25 \rightarrow 16, 35 \rightarrow 22, 45 \rightarrow 31, 55 \rightarrow 72, 65 \rightarrow 79, 75 \rightarrow 83, 85 \rightarrow 92, 95 \rightarrow 100$

These fairness concerns also highlight the problems with using conditional independence to determine bias for the weak rankers. Because it is a group fairness method, and weak rankers can have relatively low accuracy, to satisfy this metric only the proportion correctly ranked matters. This can lead to tokenization, where a few protected class elements are overranked and the rest are underranked; oscillation, where the protected class ends up in clusters due to the weak

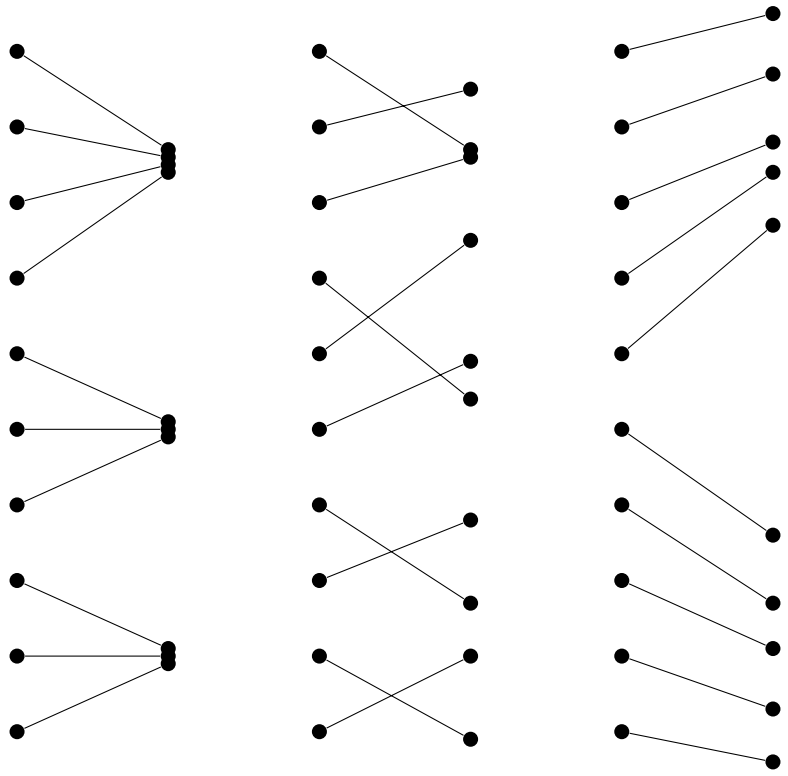

Figure 2.3. Undetected Clustering, Noisy Ranking, and Tokenization Examples

rankers labeling these elements inconsistently; or noisy ranking, where the weak rankers ignore the correct ordering of these elements, for example by causing too many ties to stratify these elements effectively.

2.2.4 FARE ranking metrics

This section will evaluate the three ranking specific fairness metrics proposed by Kuhlman, et al. since they are specifically intended to be applied to ranking problems[31]. In fact, since these were the only truly ranking specific metrics uncovered in the literature review for this section, it is imperative to test them before they become the de facto metrics. Although the FARE paper does provide some helpful background information and some similar rationale to this paper for creating metrics for ranking that are different from those used for classification, it unfortunately goes into very limited mathematical detail about the robustness of these metrics. All are

group parity metrics, and can be checked against the desirable qualities for fairness metrics explained in the previous section. It is interesting to note that the three metrics described are all computed pairwise, which is the main innovation distinguishing them from the original classification fairness metrics. This design is therefore easily incorporated into pairwise boosting algorithms for ranking such as RankBoost.

The first fairness metric proposed by the paper is called Rank Equality Error, which they define according to the following equation, where H is the ensemble ranker working on pairs x_i which have correct pairwise ordering y_i. The indicator function in the numerator is 1 when an element from group A_1 is incorrectly ranked above an element from a different group, whereas the normalizing factor in the denominator counts instances where the pairs have elements from different groups.

$$R_{eq}(H(x), y) = \frac{\sum_i |H(x_i) - y_i| \, \mathbb{I}(A_1 > A_2)}{\sum_m \mathbb{I}(A_1 \neq A_2)}$$

Essentially, this metric counts and normalizes the number of incorrect pairs where the specified group is ranked more highly than elements of different groups. According to the FARE authors this approach is based on equalized odds or conditional parity. As the authors of the paper point out, the efficacy of this metric is dependent on the true rank given by the dataset being fair, so it will not detect a smooth, uniform distribution. Metrics based on incorrect pairings will be blind to biases originating in the data. Similar to the basic version of conditional parity, this metric cannot effectively detect tokenization, although the extreme case of bimodal rankings will result in a relatively poor score for this metric. Because this metric detects the proportion of overrankings out of the total misrankings, some severe overrankings can be "balanced" by many underrankings (see Table 2.1). This metric will detect arbitrary treatment of the protected class though, since each individual element in the protected class is evaluated against all pairs against members of other classes.

Rank Equality Error on Ranking $B_9 A_0 A_2 B_1 B_3 A_5 B_4 A_8 B_6 B_7$				
	A_0	A_2	A_5	A_8
B_1		Y		
B_3				
B_4			Y	
B_6				Y
B_7				Y
B_9	X	X	X	X

Table 2.1. An Example of Rank Equality Error: both A and B have $R_{eq} = \frac{1}{6}$
A and B are two classes defined by some protected characteristic such that their treatment is expected to be similar according to the fairness criteria used.
The subscripts on each element denote the correct position of that element.
This table gives the pairs which compare an element of A to an element of B; X means that the element of B is incorrectly ranked above the element of A and Y means that the element of A is incorrectly ranked above the element of B.

The second proposal from the paper is called Rank Calibration Error which is the proportion of incorrect pairwise rankings that impact the protected class.

$$R_{cal}(H(x), y) = \frac{\sum_i |H(x_i) - y_i| \, \mathbb{I}(A_1)}{\sum_m \mathbb{I}(\neq A_1)}$$

Rank calibration is based on calibration for classification which indicates the correctness of the model's prediction for each group. Both overranking and underranking elements of the protected class reduce the rank calibration, and the two types of discordance are not recorded separately. Similarly to Rank Equality Error, this metric will be heavily dependent on the fairness of the original dataset and its true rankings, so it will not detect uniform distribution of elements of the protected class. Its main utility is to evaluate whether the protected class is being consistently ranked less precisely than other elements, which is somewhat related to the notion of cost explored in the following chapter. Because it does not distinguish between underranking and overranking, Rank Calibration Error will not detect tokenization, clustering, or other forms of systematic error as opposed to noisy rankings of the protected elements. However, it will detect severe arbitrary treatment of the protected class, since this will significantly impact the proportion of misranked pairs.

The final metric Kuhlman, et al. propose is Rank Parity Error, calculated using the following equation,

$$R_{par}(H(x), y) = \frac{\sum \mathbb{I}(A_1 > A_2)}{\sum_m \mathbb{I}(A_1 \neq A_2)}$$

which is based on statistical independence since it has appeared in several of the other papers exploring fairness for ranking[5,36,38]. Rank Parity Error describes the number of pairs where an element of the protected class is ranked higher than an element outside of the protected class, regardless of whether the rank is correctly ordered. The goal of this metric is to detect the distribution of elements of the protected class across the rank. Ideally, the rank parity value will be about $\frac{1}{2}$ for a relatively uniform spread. While the goal of this approach aligns with the desirable characteristic of being able to detect a uniform spread as described in the previous section, in a number of vital scenarios it will not be able to detect failures to achieve this fairness objective. Fundamentally it does not solve the issue of tokenization or clustering and even is unlikely to detect an extreme bimodal ranking, as shown in the example given in Table 2.2. The number of pairs where the protected class elements are ranked higher than non-protected class elements can be "fair" according to this metric through systematic underranking and overranking. Therefore, in the important benchmark scenarios laid out in this paper, this metric will be inadequate in its paper's stated goal of detecting uniform distributions.

Although the premise of introducing ranking specific versions of fairness metrics is worthy, these metrics still fall short in many ways. They provide a convenient framework for calculating properties that very closely align with those which are prevalent for fair classification tasks. However, the metrics do not do enough to improve their representativeness to the fair ranking problem. They still fail in most of the same ways which negatively impact the classification fairness metrics applied to the fair ranking problem, which demonstrates that Rank Equality

Rank Parity Error on Ranking $A_0 A_2 B_1 B_3 B_4 B_6 B_7 B_9 A_5 A_8$				
	A_0	A_2	A_5	A_8
B_1	X	X		
B_3	X	X		
B_4	X	X		
B_6	X	X		
B_7	X	X		
B_9	X	X		

Table 2.2. An Example of Rank Parity Error: both A and B have perfect $R_{par} = \frac{1}{2}$
A and B are two classes defined by some protected characteristic such that their treatment is expected to be similar according to the fairness criteria used.
The subscripts on each element denote the correct position of that element and X means that the model ranks the pair such that $A > B$.

Error, Rank Calibration Error, and Rank Parity Error for the most part merely extend the overly-simple-for-ranking classification fairness metrics. With failure to detect even basic problematic scenarios, as one would desire for a robust, generalizable fair ranking metric, the work done by the FARE paper does not substantially improve upon the flaws present in the bias metrics for classification when applied to ranking.

2.2.5 Preference-based Fairness

Preference-based fairness for classification is intended to improve or maintain the impact of a trained model for all groups as compared with parity-based fairness by relaxing the strict parity requirement[19]. Error is minimized by using classifiers for each group that provide more desirable outcomes than the parity achieving options would. Recall the linear program for classification, where l_{h_z} is the loss function on the classifier h_z preferred by class z over the impact parity classifier h'.

$$\begin{aligned} \underset{h_z}{\text{minimize}} \qquad & \frac{1}{N} \sum_{(\mathbf{x},y,z)} l_{h_z}(\mathbf{x},y) + \sum_{z \in \mathcal{Z}} \lambda_z \Omega(h_z) \\ \text{subject to} \qquad & \sum_{\mathbf{x} \in \mathcal{D}} \max\left(0, h_z^T \mathbf{x}\right) \geq \sum_{\mathbf{x} \in \mathcal{D}_z} \max\left(0, h'^T \mathbf{x}\right) \text{ for all } z \in \mathcal{Z} \end{aligned}$$

This concept is appealing for ranking fairness due to its higher flexibility which could make the simultaneous optimization of performance and fairness result in a smaller tradeoff. Conceptualizing a preferred ranking scheme requires a single metric to determine which ranking models are more desirable for a group, but can be approximated by allowing each group to "choose" weak rankers to add to the ensemble for a boosted ranking algorithm.

$$\underset{\mathbf{h}_{|z|} \in \mathscr{H}}{\text{minimize}} \qquad \hat{E}_2(\mathbf{h}_{|z|}, \mathbf{x}, y)$$

$$\text{subject to} \qquad \sum_{x_a \in \mathscr{D}_z, x_b \notin \mathscr{D}_z} h^T(x_a) - h^T(x_b) \geq \sum_{x_a \in \mathscr{D}_z, x_b \notin \mathscr{D}_z} h'^T(x_a) - h'^T(x_b) \text{ for each } z \in \mathscr{Z}$$

Here, the performance optimization takes the exponential rank loss across the newly selected binary weak rankers from each group, and the group benefit constraint considers the treatment of the individual members rather than the pairs by the candidate weak rankers. Given the possibilities, each group will select the weak ranker that elicits the most substantial benefit for their group as compared to the other groups. However, integrating multiply rankers for each class introduces new challenges and the need to weight the weak rankers mean this approach is fraught with potential for exploitation. For the proposal above, each set of $|z|$ weak rankers would have to be weighted evenly together according to the aggregate error of their mini-ensemble.

Observation 3. Preference-based techniques are sensitive to class asymmetry in the training data and moreover require a reliable parity based metric against which to compare performance to rate the quality of rankers.

Proof: Assume there exists an established parity metric for weak rankers, and that groups have a success metric which deterministically selects a weak ranker that most benefits the group as a whole. Consider an ensemble ranker composed of weak rankers selected by three different groups, Tories (T), Labour (L), and Lib-Dems (D), each acting in its own interest to maximize its ranking success. Assume the true ranking of the candidates is $T_1, L_2, D_3, T_4, T_5, T_6, D_7, L_8, T_9$. Labour and the Lib-Dems each select a weak ranker h_L and h_D that values their respective

members as 1 and all other candidates as 0. Meanwhile, the Tories pick a weak ranker h_T that assigns a 1 to T_1, T_4, T_5, and T_6, and values every other candidate as 0. Each of these weak rankers performs better for the group that selected it than a parity-based metric would, which incentivizes its selection. However, when these weak rankers have to be weighted for the final ensemble, this most benefits the Tories, since they are the group with plurality which can minimize loss by correctly ordering their own members with a weak ranker. Both h_L and h_T misrank 7 pairs, whereas h_T misranks only 6. Therefore, h_T will be receive the highest weight among the three rankers, which means that T_4, T_5, and T_6 will receive more of the vote share than either L_2 or D_3, which each receive one vote from a lower weighted weak ranker. This example helps to illustrate the potential for "gerrymandering" when transforming preference-based classification to preference based ranking[30]. Ultimately, the groups will have chosen mutually contradictory weak rankers to benefit their own members, so the most influence will default to the group whose majority population allows the greatest degree of self-promotion compliant with the canonical rank of the training data.

2.2.6 Sub-group Fairness and Individual Fairness

Subgroup fairness for ranking is also dependent on having a robust group parity metric, which is currently lacking as demonstrated above. To determine subgroup treatment is to recursively compute group treatment parity on intersectional or increasingly precise subgroups. Without a meaningful comparison method, these attempts will be rendered futile. Unfortunately, individual fairness, which is currently making significant headway in fair classification research, is also very difficult to extend to ranking. The fundamental notion that "similar individuals should be treated similarly" becomes much harder with the full preferential ordering required for ranking as compared to the binary separation involved in classification. The entire point of ranking is that there should be an objective way to form a quality gradient, so wanting similar individuals to get similar treatment is circular. Consider inviting job applicants from the same school for

interviews; even if they have similar qualifications, and both can get invited for an interview, there is some preferential ordering which implies dissimilarity. Since the purpose of ranking is to differentiate between all elements, the degree of similarity of treatment is inherently limited. Ranked individuals can never get the same treatment, so at best every element can get its deserved position in the hierarchy, since that is what is most fair. This objective implies ranking fairness for individuals is based on the implicit quality of the trained ranker rather than an external fairness quantifier. In order to make progress in this area, work in preprocessing datasets and adjusting ranking algorithms to be more aware of the consistent ranking treatment of individuals in the protected class may be necessary. In the following chapter, adjustments to make RankBoost cost-sensitive could be a step forward in this direction, as giving a weighted sense of the importance of precise ranking to certain elements may be able to improve the performance of the trained model on the protected class elements.

2.3 Bias Mitigation for Multiclass Classification

Bias mitigation for multiclass classification is a different problem from fair ranking, even at the limit, because there is not an explicit assignment of preference, meaning it can be thought of a horizontal rather than vertical stratification of elements. However, multiclass classification exists in the space between binary classification and ranking due to the increased complexity when the mutual relationships between more than two group labels must be considered. In particular, this makes class asymmetry a central consideration for many multiclass classification problems. There has been some work in weighted (cost-aware) algorithms for multiclass classification which provides a useful point of comparison in the following chapter[14,18].

Multiclass classification also raises the issue of one-vs-all and all-vs-all approaches, which has been compared to fairness problems in binary classification[20]. One-to-all approaches choose a class, generally the class with plurality, and perform pairwise comparisons between that class

and each of the other classes[41]. Effectively, this technique relies on turning the multiclass problem into a centralized separator with the smaller classes being spokes out from the biggest class. This strategy naturally leads to asymmetric classification challenges where separation between non-majority classes is systematically biased towards the features of the majority class, which is also likely to be overrepresented in the results[42]. As a result, other techniques have been explored with complete pairwise comparison between all classes. While fairness is not explicitly the aim, improving performance by targeting systematic bias in a more complex system than binary classification can help to inform the goals for both fair multiclass classification and fair ranking.

2.3.1 Fair Multiclass Classification

Although most specific research on fairness for multiclass classification relates to creating explainable neural networks[43–45], because of the connection between n-label multiclass classification and ranking it is worth exploring how the fair binary classification metrics can be extended to the fair multiclass classification problem. Statistical independence can be immediately applied to k-multiclass classification problems, where $k << n$, without needing to distinguish between one-vs-all or all-vs-all comparison. Its requirement is simply that being a member of the protected class does not statistically impact what classification an element receives from the model, so all the labels would need to share the same proportion of members of the protected class. This proportionality can be established from the majority class and applied in pairwise comparisons to the minority classes without necessitating complete pairwise interaction. Other approaches involve increased nuance due to the multifaceted connections between different classification groups. However, it is important to note that as the groups become increasingly fine-grained, i.e. as the multi-class classification approaches n-label separation, the approach described above ceases to have meaning since their is no proportion in a one-element group. At this limit, statistical independence runs into the challenges described in 2.2.2 where

the expectation is insufficient to detect the spread of the elements across the n-labels.

Consider conditional independence for a multiclass classification problem, which depends on symmetry in the binary classification problem. Should the expected label of misclassified elements of the protected class be proportionately distributed across all incorrect groups? Or can all misclassifications be considered equal? Does similarity between the groups matter when considering the penalty or degree of unfairness for misclassification? Ensuring misclassifications are identically distributed for protected and non-protected class elements is arguably counterproductive, but the potential for bias is clear. This challenge is exacerbated for subgroup fairness due to the increased fragmentation from the binary classification problem.

One possibility is to weight the multiclass labels pairwise based on their problem-specific "distance" from each other, and distribute misclassifications according to this mapping. This idea extends naturally from an intuitive notion of individual fairness for a multiclass problem, where each element can be considered to be in hyperdimensional space which should be separated equitably for all individual elements. A misclassification in a related or nearby group would in this case be considered less severe than a more grevious misclassification, while allowing a less rigid system for evaluating the fairness of misclassifications. However, the problem specific constraints make it difficult to meaningfully extend this approach to the analogous ranking problem.

3 Cost-Sensitivity for Ranking

This chapter begins with 3.1, which describes Nikolaou's dissertation work to unify the many cost-sensitive versions of AdaBoost and gives a framework for evaluating the theoretical properties of their loss functions[1]. In the subsequent section, 3.2, the paper introduces a cost-sensitive RankBoost variant which can be extended to similar algorithms such as RankBoost+ (3.2.1)[11,46]. This section also includes proofs about the loss functions and weighting for weak rankers for both cost-sensitive variants. 3.3 provides proofs about further theoretical properties, including the cost-sensitive generalization bound (3.3.2). This research effort to create a robust cost-sensitive approach for boosted ranking algorithms is based on the potential applications to bias mitigation for ranking, such as the use of cost-sensitive algorithms to create fair classifiers[29] and the work from 2.2.1 and 2.2.6 which imply that more precise ranking of the protected class given preprocessed training data can improve a number of measures for fairness outcomes. Since cost is a way to express average risk, and bias is inherently a risk of unfairness that needs to be mitigated, having a framework for cost-sensitive ranking provides the necessary foundation to transform fairness problems in ranking to cost-sensitive ranking.

3.1 Cost-Sensitive Boosted Classification Algorithms

In many machine learning problems, different mistakes have consequences of different magnitudes. Barring a perfect model, there will be some tradeoff between the model's ability to classify or rank elements along the margin or within a protected or rare class. In these asymmetric

scenarios, where one class has fewer instances than the other or where the cost of mislabeling differs, leading to an asymmetric confusion matrix, algorithm modifications can train models that better capture the consequences of asymmetry[1]. In classification, this cost or class asymmetry generally reflects the difficulty of assigning a relatively less common label or the risk of a false positive or false negative in a sensitive scenario such as medical testing. For classification, many cost-sensitive algorithms and variants have been proposed to address this need. For the purposes of this paper, the focus will be on boosting algorithms, which either explicitly or implicitly modify the loss function of the algorithm in order to better capture the asymmetry of the problem in the model. Using transformations between cost and class asymmetry, it can be shown that the two problem categories are mathematically equivalent and can be addressed using the same algorithmic techniques.

3.1.1 AdaBoost and its Cost-Sensitive Variants

AdaBoost[46], short for "Adaptive Boost," helped to originate the use of weighted voting systems for ML classification problems. By aggregating weak learners, h_t, as long as they perform better than guessing, it is possible to combine relatively poorly performing classifiers into a much more powerful ensemble. Weak rankers are incorporated into the weighted voting model according to the combination of voters which minimizes some loss metric on the training data. Based on computability concerns, the process of picking weak learners is done iteratively instead of by optimizing to the true minimum, but both empirical results and theoretical analysis have affirmed the success of this strategy. The final weighting, α_t, of each weak learner h_t is calculated according to its weighted error, ϵ_t.

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

The distribution of the N labeled training examples (x_i, y_i) is initially assigned uniform weight,

$$D_i^0 = \frac{1}{N},$$

where D is the normalized weight distribution, in this case for pair i at iteration 0. The weight distribution of all the pairs is adjusted at each step according to the performance of the current ensemble with the newly incorporated weak learner using the update step below, where Z_t is the normalization constant for the reweighted distribution.

$$D_i^{t+1} = \frac{D_i^t \times e^{-y_i \alpha_t h_t(x_i)}}{Z_t}$$

The cost-sensitive variants of AdaBoost have used a variety of approaches to incorporate a notion of cost into the boosting algorithm, as seen in Figure 3.1.

Nikolaou provides a thorough analysis of the properties and relative merits of these cost-sensitive AdaBoost variants according to the loss-function properties outlined in the following section [1]. He groups these algorithms into those that modify the training step, those that adjust D^0, and those that calibrate the classification differently in post-processing. According to his analysis, which calculated the effective loss functions of even the ad-hoc approaches, the theoretically most sound versions applied cost as a coefficient to the loss function. The three algorithms that utilized this approach did so using the initial distribution (CGAda), during the weight update rule (AsymAda) [47], and through post-processing the normal AdaBoost algorithm, respectively. These distinctions led to slightly different empirical results, but resulted in the minimization of the same loss function, $c(y)e^{-yF_t(x)}$, where c gives the pairwise cost and F_t is the classifier at iteration t. The differences in results are due to the effects of boosting differing between the versions due to when the cost is applied, but theoretically these three approaches are identical and the starting point for the work on cost-sensitive ranking described in this chapter.

Algorithm	**Weight Update Rule** $D_i^{t+1} \propto [\ldots] \times D_i^t$	**Initial Weights** D_i^1 **& Cost Adjustment Functions**
AdaBoost	$e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = \frac{1}{N}$
CGAda	$e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AdaC1 CSAda AdaDB	$e^{-c(y_i) y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AdaC2	$c(y_i) e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AdaC3	$c(y_i) e^{-c(y_i) y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$
AsymAda	$c(y_i)^{1/M} e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)^{1/M}$ (fixed M)
CSB0	γ_t^i	where $D_i^1 = c(y_i)$
CSB1	$\gamma_t^i e^{-y_i h_t(\mathbf{x}_i)}$	and $\gamma_t^i = \begin{cases} c(y_i), & \text{if } h_t(\mathbf{x}_i) \neq y_i \\ 1, & \text{if } h_t(\mathbf{x}_i) = y_i \end{cases}$
CSB2	$\gamma_t^i e^{-y_i \alpha_t h_t(\mathbf{x}_i)}$	
AdaCost AdaCost(β_2)	$e^{-\beta_t^i y_i \alpha_t h_t(\mathbf{x}_i)}$	where $D_i^1 = c(y_i)$ and $\beta_t^i = \begin{cases} \frac{1+c(y_i)}{2}, & \text{if } h_t(\mathbf{x}_i) \neq y_i \\ \frac{1-c(y_i)}{2}, & \text{if } h_t(\mathbf{x}_i) = y_i \end{cases}$

Figure 3.1. Cost-Sensitive AdaBoost Variations Analyzed by Nikolaou[1]

3.1.2 Evaluating Loss-Functions

Nikolaou introduces four metrics by which to consider the properties of loss-functions presented by cost-sensitive variants of AdaBoost[1]. The behaviors of the resulting models reflect the extent to which each variant obeys the requirements of functional gradient-descent, decision theory, margin theory, and probability theory.

- Functional gradient-descent minimizes the loss on the training data incrementally by incorporating new weak learners that progressively lower the error of the model according to the functional derivative. The weight update rule, which determines the

new weight distribution of the elements in the training data, depends on the nature of the loss function and its functional derivative. Picking the best weak learner means taking the greedily optimal loss minimizing step according to $\frac{\partial L(yF(x))}{\partial yH(x)}$, where L is the loss function on the correct label y and the label chosen by the model $F(x)$. Modifications to the weight update rule are evaluated based on whether this loss optimization step is still locally optimal.

- The decision-theoretic approach builds on the functional gradient-descent premise from above but focuses on the construction of the minimizer H. The minimizer should be optimal by the final round of boosting according to the cost assigned to each type of mistake. Nikolaou uses the label "cost-consistent" for algorithms that apply their minimizier model to classify elements according to the costs and the label "cost-inconsistent" for those that classify at a threshold other than costs. This property may seem obvious, but was rendered necessary due to the large number of heuristic-based cost-sensitive AdaBoost variants that were found to be cost-inconsistent.
- Margin theory can be applied to cost-sensitive boosting by introducting the concept of "asymmetry preservation." For a given margin, elements that have higher cost should always be prioritized, i.e. by having higher penalty for misclassification, than the elements with the same margin but from the lower cost class. This strategy is based on research that indicates that increasing margins in the model leads to better generalization [48,49].
- The probabilistic view as described by Nikolaou points out that the confidence values produced by boosting cannot immediately be used as probability estimates. The extremity of division by boosting necessitates post-processing calibration in all explored cost-sensitive AdaBoost variants in order to turn the output for elements into a calibrated probability score. This property potentially connects to the use of calibration

as a fairness metric discussed earlier, since post-processing output to generate proper probabilities can stabilize or smooth the behavior of the model.

Unfortunately, these metrics cannot all immediately be translated to apply to cost-sensitive ranking. They nonetheless provide a useful starting point to consider the priorities for the properties of a cost-sensitive variant of a boosting algorithm for ranking, and help to highlight the similarities and differences between the problems of cost-sensitive classification and cost-sensitive ranking.

3.2 Cost-Sensitive RankBoost

RankBoost is similar to AdaBoost in many ways, but differs in the computation of the loss function during training. AdaBoost's rank loss function is based on misclassifications as shown below, with $h(x)$ being the classifier h's label for element x, y being the correct label for x, and D^t being the weighting distribution on the elements at the iteration t,

$$\hat{R} = \sum_{i=1}^{n} h(x_i) \neq y_i \; D_i^t$$

In contrast, the RankBoost rank loss function uses pairwise comparisons between elements to determine which pairs are correctly ordered and penalizes those that are swapped[4]. The pairs that were ranked incorrectly are reweighted for the next round of boosting so that future weak rankers will be incorporated to enforce the correct ordering between those pairs. Therefore, to incorporate cost-sensitivity into RankBoost, the costs must either be assigned to the pairs, or be assigned to the elements in such a way that the pairwise cost can be meaningfully computed for each pair of elements. The cost-structure for this RankBoost variant is based on Nikolaou's conclusions about the performance of cost-sensitive AdaBoost variants, which indicated that only the algorithms that applied cost as a coefficient achieved all of the desirable theoretical properties described above[1]. Hence, this cost-sensitive RankBoost variant also applies cost as

a coefficient such that the new Rank Loss function is

$$\hat{\mathbf{R}}_2 = \sum_{i=1}^{m} \tilde{c}(i)e(i)$$

where the loss is summed over m pairs, $\tilde{c}(i)$ is the normalized cost of the i^{th} pair, and $e(i)$ is an indicator function that denotes if the result was incorrect with value $\frac{1}{2}$ assigned to ties. Note that in this paper, the cost sensitive loss functions are bolded to distinguish them from the loss functions of the original algorithms. The same coefficient appears in the exponential loss function that is used to optimize the ranking model during training, as shown in the following equation.

$$\hat{\mathbf{E}}_1\left(\sum_{s=1}^{N} \eta_s f_s\right) = \frac{1}{m}\sum_{i=1}^{m} \tilde{c}(i)e^{\sum_{s=1}^{N} \ln \omega_s^0(i,f_s,\eta_s)}$$

Note that because the "decision rule" (final ordering of the ranking) of the loss minimizer is a function of the ensemble ranker's error and the cost function, this incorporation of cost into the ranking algorithm is cost-consistent. This cost function can be incorporated as the initial distribution instead of starting with uniform weight, applied evenly across each iteration of the training where pairs are reweighted by $c(x_i)^{1/M}$ for each of M iterations, or as a post-processing step where the output of the model is adjusted by the cost. Because of the theoretical equivalence of these techniques, disregarding small empirical differences, these approaches are effectively identical. For the purpose of simplicity, cost will be incorporated in the initial distribution in the following sections.

Cost-sensitive RankBoost should still satisfy the first three quality metrics described in the previous section for cost-sensitive classification. Probabilistic calibration is not immediately useful but could be an interesting area of future work. The ensemble ranker is constructed by greedily optimizing a loss function with the incorporation of predictive weak rankers. This can be structured as the construction of a loss minimizer according to the decision theoretic perspective. Margin theory is slightly more complicated due to the full ordering of the elements rather than

their binary separation, but ranking margins can also impact the generalization quality of the model. The notion of asymmetry preservation is particularly apt for the ordering of elements, as correct ordering should be prioritized for higher cost elements given the same degree of separation as lower cost elements. With the computation of the loss function for cost-sensitive RankBoost in later sections, revisiting these concepts will provide further insight into the efficacy of this cost structure for boosted ranking algorithms.

3.2.1 RankBoost+

RankBoost+ challenges the treatment of ties in the original RankBoost algorithm, asserting that ties should be reweighted by the mean of the reweighting for correctly and incorrectly ordered pairs[11]. Because in RankBoost ties are treated as incorrect in $\hat{R}_1$ but weighted as 1 in $\hat{E}_1$,

$$\hat{R}_1(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(y_i(h(x_i^{'}) - h(x_i)) \leq 0)$$

$$\hat{E}_1(h) = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i(h(x_i^{'}) - h(x_i))}$$

the rank loss and the exponential loss functions produce different orderings and weighting of rankers. Therefore, using $\hat{E}_1$ as an optimizing approximation of the rank loss may be detrimental to the model being trained in terms of ability to minimize $\hat{R}_1$. Furthermore, the exponential loss function weights ties in such a way that they are not penalized proportionately to the reweighting of correct versus incorrect pairs. RankBoost+ suggests an intervention to adjust the reweighting function to

$$\omega_2^*(i, f_s, \eta_s) = \begin{cases} e^{-\eta_s} & \text{if pair } i \text{ correctly ranked by } f_s \\ e^{-\eta_s} & \text{if pair } i \text{ is incorrectly ranked by } f_s \\ cosh(\eta_s) & \text{if pair } i \text{ is tied by } f_s \end{cases}$$

which results in minimizing the exponential loss function $\hat{E}_2$, based on the rank loss function $\hat{R}_2$ used to evaluate performance in the original RankBoost paper[4]. With ties weighted halfway

in between an incorrect and a correct pairwise ranking, for binary weak rankers incorporated iteratively there is no longer any reversal of ranker ordering between rank loss and its exponential approximation. The exponential loss function on the ensemble, with α_s as the weight of h_s, up to the current iteration t where the ranker h_t is incorporated with weight h_t, is

$$\hat{E}_2\left(\sum_{s=1}^{t}\alpha_s h_s\right) = \frac{1}{m}\sum_{i=1}^{m} e^{\sum_{s=1}^{N}\log\omega_2^*(i,f_s,\eta_s')}$$

where η_s' is the vector which describes the weighting of the weak rankers at the time step being computed. Ultimately, the new α_t value best encapsulates the impact of this adjustment to the reweighting scheme during optimizations,

$$\alpha_t = \frac{1}{2}\log\left(\frac{\epsilon_t^{+1} + \epsilon_t^0\frac{e^{-\alpha_t'}}{2cosh(\alpha_t')}}{\epsilon_t^{-1} + \epsilon_t^0\frac{e^{\alpha_t'}}{2cosh(\alpha_t')}}\right)$$

which uses the exponential expansion of the hyperbolic cosine function to balance the weighting of rankers according to the new policy towards tied pairs. Here,

$$\epsilon_t^{\tau} = \sum_i D_i^t\,\mathbb{I}([h_t(x_i^{'}) - h_t(x_i)] == \tau)$$

where $\tau \in \{-1, 0, +1\}$ indicates the pairs which are reversed, tied, and correct. The cost-sensitivity approach described in the previous section can naturally be extended to RankBoost+ without changing its desirable properties. Both versions will be used going forward and their performances will be compared in the experimental sections; however proofs which are interchangeable between the versions will not be duplicated.

3.2.2 Loss Functions

Proving the basic properties of Cost-sensitive RankBoost and RankBoost+ demonstrates the similarities and differences between the two variations as well as their distinguishing features as compared to the standard (non-cost aware) versions of the algorithms. The performance and optimization criterion are based on the new loss functions for each version, and the characteristics of those loss functions are proved throughout this section as necessary.

Algorithm 1 Pseudocode for Cost-Sensitive RankBoost and RankBoost+

function COST-SENSITIVE RANKBOOST($(x_1, x_1', y_1, \tilde{c}_1), ..., (x_m, x_m', y_m, \tilde{c}_m)$)
  for $i = 1$ to m **do**
    $D_0(i) = \tilde{c}_i$
  for $t = 1$ to T **do**
    $h_t = \arg\min_{h \in \mathscr{H}'} |\delta(h)|$ for RankBoost or $h_t = \arg\max_{h \in \mathscr{H}'} |\delta(h)|$ for RankBoost+
    $\alpha_t = \frac{1}{2} \log \frac{\epsilon_t^+}{\epsilon_t^-}$ for RankBoost or $\alpha_t = \frac{1}{2} \log \left(\frac{\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha_t'}}{2cosh(\alpha_t')}}{\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha_t'}}{2cosh(\alpha_t')}} \right)$ for RankBoost+
    for $i = 1$ to m **do**
      $D_{t+1}(i) = \frac{D_t(i)\omega_t^+(i)}{Z_t}$
  $g = \sum_{t=1}^{T} \alpha_t h_t$
  return g

Theorem 1. The cost-sensitive exponential loss function $\hat{\mathbf{E}}_2$ is the same whether cost is applied before, after, or during training.

$$\hat{\mathbf{E}}_2^N = c_i \prod_{t=1}^{N} b_t = \prod_{t=1}^{N} b_t \, c_i^{1/N}$$

Call the error vector incorporated at each step b_t, where t is the current step. The cost function for each pair i is $c(i)$, and the cost vector for all pairs is written here as c_i. N is the total number of rankers currently incorporated into the ensemble.

In other words, $\hat{\mathbf{E}}_2$ can be modeled as distributing the cost across every weak ranker incorporated into the ensemble when the cost is introduced completely at the first step or after the ranker has been trained (see Lemma 1). The loss function equivalence for introducing cost in the initial distribution versus at each iteration versus during post-processing means that the theoretical properties of all three approaches will be the same, although empirical performance may differ between the implementations.

PROOF.

This proof uses an induction argument. First, assume $\mathbf{E}_2^N$ is of the form

$$\mathbf{E}_2^N = \frac{1}{m}\sum_{i=1}^{m} e^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}})}$$

after adding the N^{th} weak ranker by averaging the error across all pairs with a cost coefficient.

Next, consider the value of $\mathbf{E}_2$ once the $N+1^{th}$ ranker is added to the ensemble.

$$\mathbf{E}_2^{N+1} = \frac{1}{m}\sum_{i=1}^{m} e^{\sum_{s=1}^{N+1} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N+1}})}$$

This equation gives the product of $\mathbf{E}_2^N$ and the distribution update rule.

$$\mathbf{E}_2^{N*x} = \mathbf{E}_2^{N+1}$$

To isolate the distribution update rule, solve for the quotient x.

$$\frac{1}{m}\sum_{i=1}^{m} xe^{\sum_{s=1}^{N} \ln\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}}} = \mathbf{E}_2^{N+1}$$

$$\frac{1}{m}\sum_{i=1}^{m} xe^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}})} = \frac{1}{m}\sum_{i=1}^{m} e^{\sum_{s=1}^{N+1} \ln\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N+1}}}$$

Because cost is applied to each pair, consider w.l.o.g. some pair and its cost between iterations.

$$xe^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}})} = e^{\sum_{s=1}^{N+1} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N+1}})}$$

Isolate the last step of the summation on the right-hand-side of the equation.

$$\begin{aligned} xe^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}})} &= e^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N+1}})} e^{\ln(\omega_2^*(i,f_{N+1},\eta_{N+1})c(i)^{\frac{1}{N+1}})} \\ &= e^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}\frac{N}{N+1}})} e^{\ln(\omega_2^*(i,f_{N+1},\eta_{N+1})c(i)^{\frac{1}{N+1}})} \end{aligned}$$

Separate out the constant term in the log-exponent.

$$= e^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}}) + \ln c(i)^{\frac{1}{N}\frac{-1}{N+1}}} e^{\ln(\omega_2^*(i,f_{N+1},\eta_{N+1})c(i)^{\frac{1}{N+1}})}$$

$$= e^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s)c(i)^{\frac{1}{N}})} e^{\ln c(i)^{\frac{-1}{N+1}}} e^{\ln(\omega_2^*(i,f_{N+1},\eta_{N+1})c(i)^{\frac{1}{N+1}})}$$

Divide out the identical terms on both sides to solve for x.

$$x = e^{\ln c(i)^{\frac{-1}{N+1}}} e^{\ln(\omega_2^*(i,f_{N+1},\eta_{N+1})c(i)^{\frac{1}{N+1}})}$$

$$= e^{\ln c(i)^{\frac{-1}{N+1}} + \ln(\omega_2^*(i,f_{N+1},\eta_{N+1})c(i)^{\frac{1}{N+1}})}$$

$$= e^{\ln(\omega_2^*(i,f_{N+1},\eta_{N+1}))}$$

□

Lemma 1. Incorporating the cost during the initial distribution results in the same loss function as reweighting by cost as a post-processing step.

PROOF.

Consider the final exponential loss function of a fully trained ranker without cost incorporated.

$$\mathbf{E}_2 = \frac{1}{m}\sum_{i=1}^{m} e^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s))}$$

Compare this to the final exponential loss function of a model where the initial distribution was the cost function.

$$\mathbf{E}_2^{PRE} = \frac{1}{m}\sum_{i=1}^{m} c(i)\ e^{\sum_{s=1}^{N} \ln(\omega_2^*(i,f_s,\eta_s))}$$

Note that the only difference between these loss functions is the cost coefficient. Therefore, this cost function can be incorporated through a post-processing step. Multiply the exponential loss by the cost-function, then reweight the vote of each weak ranker by the modified exponential loss with cost-function.

$$\mathbf{E}_2^{POST} = c(i)\ \mathbf{E}_2 = \mathbf{E}_2^{PRE}$$

Although empirically the final weighted ensembles may differ based on which weak rankers were selected, the pre-processed and post-processed loss functions now match.

□

This proof, although it uses the RankBoost+ equations for the cost-sensitive algorithm, can be applied to the Cost-Sensitive RankBoost algorithm in the exact same way. The result of this property is that applying cost in the initial distribution, across each training iteration, or as a post-processing step leads to an identical loss function. Although there may be differences between the empirical results of these approaches, as suggested by Nikolaou [1], their theoretical properties will be identical. Therefore, changing the initial distribution to the cost-function for the purposes of this paper does not preclude using this model in the other two ways during implementation as the theoretical results will be consistent. In both proofs,

$$\epsilon_t^{+1} = \{D_t(i)\}_{correct}$$

$$\epsilon_t^{-1} = \{D_t(i)\}_{incorrect}$$

$$\epsilon_t^{0} = \{D_t(i)\}_{tied}$$

will be used to serve as an identity function to group pairs by how the weak ranker orders them. If both have the same score, they are tied and in ϵ^0. The treatment of this set of ties is the main distinguishing factor between the α_t values for RankBoost versus RankBoost+. The proofs below start with RankBoost because it is simpler with relation to ties, and then demonstrate that cost-sensitive RankBoost+ will also have the same α_t equation as its parent algorithm.

Theorem 2. Let α_t be the weight given by the ensemble ranker to weak ranker h_t at iteration t. For cost-sensitive RankBoost,

$$\alpha_t = \frac{1}{2}\log\frac{\epsilon_t^{-1}}{\epsilon_t^{-1}}$$

and for cost-sensitive RankBoost+,

$$\alpha_t = \frac{1}{2}\log\left(\frac{\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}}{\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}}\right)$$

Note that this result shows that each cost-sensitive algorithm variant weights weak rankers the same way as its non-cost sensitive parent algorithm.

Lemma 2. Let $\mathbf{E}_1$ be the error function minimized by cost-sensitive RankBoost, then the weight given to ranker h_t in iteration t is $\alpha_t = \frac{1}{2}\log\frac{\epsilon_t^{-1}}{\epsilon_t^{-1}}$.

Note that this matches the definition of α_t used in the original RankBoost algorithm.

PROOF.

Start with assigning costs to the initial distribution.

$$D_0 = \left\{\frac{c(i)}{\overline{c}}\right\} = \{\tilde{c}(i)\}$$

Define the reweighting rules and the weight update step, with normalizing constant.

$$\omega_t^0(i) = \begin{cases} e^{\alpha_t} \\ e^{-\alpha_t} \\ 1 \end{cases}$$

The relationship between the updated distribution and the initial distribution can be computed using the product of the normalization factors Z_s.

$$D_{t+1} = \frac{D_t(i)\omega_t^0(i)}{Z_t}$$

$$Z_t = \sum_i D_t(i)\omega_t^0(i) = \epsilon_t^{+1}e^{-\alpha_t} + \epsilon_t^{-1}e_t^{\alpha} + \epsilon_t^0$$

$$D_{t+1}(i)\prod_{s=0}^{t} Z_t = D_0(i)\prod_{s=0}^{t}\omega_s^0(i)$$

$$\sum_{i=1}^{m} D_{t+1}(i)\prod_{s=0}^{t} Z_t = \sum_{i=1}^{m} D_0(i)\prod_{s=0}^{t}\omega_s^0(i)$$

$$\prod_{s=0}^{t} Z_t = \sum_{i=1}^{m} D_0(i)\prod_{s=0}^{t}\omega_s^0(i)$$

The normalized costs can be summed out.

$$\prod_{s=0}^{t} Z_t = \sum_{i=1}^{m} \tilde{c}(i)\prod_{s=0}^{t}\omega_s^0(i)$$

Consider the weighted ensemble at time t.

$$g_t = \sum_{s=0}^{t} \alpha_s h_s$$

$$g_{t-1} = \sum_{s=0}^{t-1} \alpha_s h_s$$

$$\hat{\mathbf{E}}_1\left(\sum_{s=1}^{N} \eta_s f_s\right) = \frac{1}{m}\sum_{i=1}^{m} \tilde{c}(i)\, e^{\sum_{s=1}^{N} \ln \omega_s^0(i, f_s, \eta_s)}$$

$$\hat{\mathbf{E}}_1\left(\sum_{s=1}^{t} \alpha_s h_s\right) = \prod_{s=0}^{t} Z_s$$

Take the derivative of both sides in at time t in order to solve for α.

$$\begin{aligned}\frac{d(\hat{\mathbf{E}}_1(g_{t-1} + \alpha_t h_t))}{d\alpha_t} &= \frac{dZ_t}{d\alpha_t}\prod_{s}^{t-1} Z_s \\ &= (-\epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^{-1} e_t^{\alpha} + \epsilon_t^0)\prod_{s=0}^{t-1} Z_s\end{aligned}$$

$$\frac{d(\hat{\mathbf{E}}_1(g_{t-1} + \alpha_t h_t))}{d\alpha_t}|_{\alpha_t=0} = (-\epsilon_t^{+1} + \epsilon_t^{-1} + \epsilon_t^0)\prod_{s=0}^{t-1} Z_s$$

This is how the next weak ranker is selected.

$$h_t = \underset{h \epsilon H}{\operatorname{argmax}} \left(-\epsilon_t^{+1} + \epsilon_t^{-1} \right)$$

Set to 0 and solve for α_t

$$\frac{d(\hat{\mathbf{E}}_1(g_{t-1} + \alpha_t h_t))}{d\alpha_t} = 0$$

$$0 = -\epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^{-1} e^{\alpha_t}$$

$$\epsilon_t^{+1} e^{-\alpha_t} = \epsilon_t^{-1} e^{\alpha_t}$$

Now take the log of both sides to get rid of the exponent to solve for α_t.

$$\log \epsilon_t^{+1} - \alpha_t = \log \epsilon_t^{-1} + \alpha_t$$

$$\log \epsilon_t^{+1} - \log \epsilon_t^{-1} = 2\alpha_t$$

$$\alpha_t = \frac{1}{2} \log \frac{\epsilon_t^{-1}}{\epsilon_t^{-1}}$$

□

Lemma 3. Let $\mathbf{E}_2$ be the error function minimized by cost-sensitive RankBoost+, then the weight given to ranker h_t in iteration t is $\alpha_t = \frac{1}{2} \log \left(\frac{\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}}{\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}} \right)$.

Note that this matches the definition of α_t used in the original RankBoost+ algorithm.

PROOF.

Start with assigning costs to the initial distribution.

$$D_0 = \left\{ \frac{c(i)}{\overline{c}} \right\} = \{\tilde{c}(i)\}$$

Define the reweighting rules and the weight update step, with normalizing constant.

$$\omega_t^+(i) = \begin{cases} e^{\alpha_t} \\ e^{-\alpha_t} \\ \frac{cosh(\alpha_t+\alpha'_t)}{cosh(\alpha'_t)} \end{cases}$$

The relationship between the updated distribution and the initial distribution can be computed using the product of the normalization factors Z_s.

$$D_{t+1} = \frac{D_t(i)\omega_t^+(i)}{Z_t}$$

$$Z_t = \sum_i D_t(i)\omega_t^+(i) = \epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^{-1} e_t^{\alpha} + \epsilon_t^0 \frac{cosh(\alpha_t + \alpha'_t)}{cosh(\alpha'_t)}$$

$$D_{t+1}(i) \prod_{s=0}^{t} Z_t = D_0(i) \prod_{s=0}^{t} \omega_s^+(i)$$

$$\sum_{i=1}^{m} D_{t+1}(i) \prod_{s=0}^{t} Z_t = \sum_{i=1}^{m} D_0(i) \prod_{s=0}^{t} \omega_s^+(i)$$

$$\prod_{s=0}^{t} Z_t = \sum_{i=1}^{m} D_0(i) \prod_{s=0}^{t} \omega_s^+(i)$$

The normalized costs can be summed out.

$$\prod_{s=0}^{t} Z_t = \sum_{i=1}^{m} \tilde{c}(i) \prod_{s=0}^{t} \omega_s^+(i)$$

Consider the weighted ensemble at time t.

$$g_t = \sum_{s=0}^{t} \alpha_s h_s$$

$$g_{t-1} = \sum_{s=0}^{t-1} \alpha_s h_s$$

$$\hat{\mathbf{E}}_2(\eta) = \sum_{i}^{m} \tilde{c}(i) e^{\sum_{s=0}^{N} \ln \omega_s^+(i, f_s, \eta_s)}$$

$$\hat{\mathbf{E}}_2\left(\sum_{s=1}^{t} \alpha_s h_s\right) = \prod_{s=0}^{t} Z_s$$

Take the derivative of both sides in at time t in order to solve for α.

$$\begin{aligned}
\frac{d(\hat{\mathbf{E}}_2(g_{t-1} + \alpha_t h_t))}{d\alpha_t} &= \frac{dZ_t}{d\alpha_t} \prod_{s}^{t-1} Z_s \\
&= \left(-\epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^{-1} e_t^{\alpha} + \epsilon_t^0 \frac{sinh(\alpha_t + \alpha_t')}{cosh(\alpha_t')}\right) \prod_{s=0}^{t-1} Z_s \\
\frac{d(\hat{\mathbf{E}}_2(g_{t-1} + \alpha_t h_t))}{d\alpha_t}|_{\alpha_t=0} &= \left(-\epsilon_t^{+1} + \epsilon_t^{-1} + \epsilon_t^0 \frac{sinh(\alpha_t')}{cosh(\alpha_t')}\right) \prod_{s=0}^{t-1} Z_s
\end{aligned}$$

This is how the next weak ranker is selected.

$$h_t = \underset{h \epsilon H}{\arg\max} \left(-\epsilon_t^{+1} + \epsilon_t^{-1} + \epsilon_t^0 \frac{sinh(\alpha_t')}{cosh(\alpha_t')}\right)$$

Set to 0 and solve for α_t

$$\frac{d(\hat{\mathbf{E}}_2(g_{t-1} + \alpha_t h_t))}{d\alpha_t} = 0$$

$$0 = -\epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^{-1} e^{\alpha_t} + \epsilon_t^0 \frac{sinh(\alpha_t + \alpha_t')}{cosh(\alpha_t')}$$

Use the definitions of the hyperbolic trig functions to expand in order to cancel out terms.

$$\begin{aligned}
0 &= -\epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^{-1} e^{\alpha_t} + \epsilon_t^0 \frac{\frac{1}{2}\left(e^{(\alpha_t + \alpha_t')} - e^{(-\alpha_t - \alpha_t')}\right)}{cosh(\alpha_t')} \\
&= -\epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^0 \frac{\frac{1}{2}\left(-e^{(-\alpha_t - \alpha_t')}\right)}{cosh(\alpha_t')} + \epsilon_t^{-1} e^{\alpha_t} + \epsilon_t^0 \frac{\frac{1}{2}\left(e^{(\alpha_t + \alpha_t')}\right)}{cosh(\alpha_t')} \\
&= -\epsilon_t^{+1} e^{-\alpha_t} + \epsilon_t^0 \frac{-e^{-\alpha_t} e^{-\alpha_t'}}{2cosh(\alpha_t')} + \epsilon_t^{-1} e^{\alpha_t} + \epsilon_t^0 \frac{e^{\alpha_t} e^{\alpha_t'}}{2cosh(\alpha_t')}
\end{aligned}$$

$$= e^{-\alpha_t}\left(-\epsilon_t^{+1} - \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}\right) + e^{\alpha_t}\left(\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}\right)$$

$$e^{-\alpha_t}\left(\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}\right) = e^{\alpha_t}\left(\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}\right)$$

Take the natural logarithm of both sides in order to cancel out the exponent.

$$\log\left(e^{-\alpha_t}\left(\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}\right)\right) = \log\left(e^{\alpha_t}\left(\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}\right)\right)$$

$$-\alpha_t + \log\left(\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}\right) = \alpha_t + \log\left(\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}\right)$$

$$\log\left(\frac{\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}}{\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}}\right) = 2\alpha_t$$

$$\frac{1}{2}\log\left(\frac{\epsilon_t^{+1} + \epsilon_t^0 \frac{e^{-\alpha'_t}}{2cosh(\alpha'_t)}}{\epsilon_t^{-1} + \epsilon_t^0 \frac{e^{\alpha'_t}}{2cosh(\alpha'_t)}}\right) = \alpha_t$$

□

Explainability is important, especially when implementation may deviate from the theoretical properties due to computability concerns such as when training a ranking model. Demonstrating that the weighting process is the same for the cost-sensitive variants supports the assertion that the cost paradigm introduced in this paper works by modifying the weak ranker selection. The premise of the cost-sensitive variants is to provide different prioritization to element pairs based on the relative importance of their being ranked correctly. This should not impact how rankers are weighted with respect to their performance, demonstrating that the cost-sensitive variants very closely follow the properties of the original algorithms.

3.3 Properties of Cost-Sensitive RankBoost

This section will introduce properties of cost-sensitive RankBoost and RankBoost+ and prove their consistency with the original versions of the algorithms given the inclusion of cost. The empirical tests will provide an opportunity to evaluate the performance of the cost-sensitive modifications as compared to the original versions of the algorithms according to these properties. The differences in theoretical behavior can be observed through a variety of cost-sensitive experiments which push different facets of this technique. Using random cost, ranking for multiclass classification, and existing cost-analogous datasets will demonstrate the comparative performance across a swath of different applications which supports the theoretical differences and intended functionality of the algorithm variants.

3.3.1 Consistent Ordering between $\hat{E}_2$ and $\hat{R}_2$ for Cost-Sensitive RankBoost+

Connamacher, et al. suggest that consistent ordering between $\hat{E}_2$ and $\hat{R}_2$ is a desirable property for boosting algorithms for ranking[11]. This property applies when the exponential loss and the rank loss always induce the same ordering on the rankers given the initial distribution of the elements. Standard RankBoost, and its cost-sensitive variant, do not abide by this property in all cases, leading to the potential of some reversed pairs between the rank loss and the exponential loss functions. The proof that the exponential loss function orders weak rankers the same as the rank loss function for cost-sensitive RankBoost+ demonstrates that the cost-sensitivity intervention preservers this intuitively beneficial property.

Theorem 3. For any pair of weak rankers h_1, h_2, if $\hat{R}_2(h_1) > \hat{R}_2(h_2)$, then $\hat{E}_2(h_1) > \hat{E}_2(h_2)$.

PROOF.

Assume we are working with binary weak rankers h_x, such that the rankers "succeed" at ranking the elements more than half of the time, leading to $\hat{R}_2(h_x) < \frac{\hat{c}(i)}{2}$. Otherwise, flip the ranker to achieve this, since the opposite ranking will result in $\hat{c} - \hat{R}_2(h_x)$.

$$\hat{R}_2(i) = \begin{cases} \tilde{c}(i) & \text{if incorrect} \\ \frac{\tilde{c}(i)}{2} & \text{if tied} \\ 0 & \text{else} \end{cases} \qquad e(i) = \begin{cases} 1 & \text{incorrect} \\ \frac{1}{2} & \text{tied} \\ 0 & \text{correct} \end{cases}$$

$$\hat{R}_2(h_i) = \sum_{i=1}^{m} \tilde{c}(i) e(i)$$

$$\hat{E}_2(h_i) = \sum_{i=1}^{m} \tilde{c}(i) \omega^*(i)$$

$$\omega^*(\alpha) = \begin{cases} e^{-\alpha_t} & \text{correct} \\ e^{\alpha_t} & \text{incorrect} \\ cosh(\alpha_t) & \text{tied} \end{cases}$$

$$\begin{aligned} \hat{R}_2(\alpha_t h_t) &= \sum_{i=1}^{m} \tilde{c}(i) \left(\frac{\omega^*(i) - e^{-\alpha_t}}{e^{\alpha_t} - e^{-\alpha_t}} \right) \\ &= \frac{\hat{E}_2(\alpha_t h_t) - \sum_{i=1}^{m} e^{-\alpha_t} \tilde{c}(i)}{e^{\alpha_t} - e^{-\alpha_t}} \\ \hat{R}_2(\alpha_1 h_1) &= \frac{\hat{E}_2(\alpha_1 h_1) - \sum_{i=1}^{m} e^{-\alpha_1} \tilde{c}(i)}{e^{\alpha_1} - e^{-\alpha_1}} \\ \hat{R}_2(\alpha_2 h_2) &= \frac{\hat{E}_2(\alpha_2 h_2) - \sum_{i=1}^{m} e^{-\alpha_2} \tilde{c}(i)}{e^{\alpha_2} - e^{-\alpha_2}} \end{aligned}$$

Assume $\hat{R}_2(\alpha_1 h_1) < \hat{R}_2(\alpha_2 h_2)$.

$$\frac{\hat{E}_2(\alpha_1 h_1) - \sum_{i=1}^{m} e^{-\alpha_1} \tilde{c}(i)}{e^{\alpha_1} - e^{-\alpha_1}} < \frac{\hat{E}_2(\alpha_2 h_2) - \sum_{i=1}^{m} e^{-\alpha_2} \tilde{c}(i)}{e^{\alpha_2} - e^{-\alpha_2}}$$

First consider $\alpha_1 = \alpha_2 = 1$. In this case:

$$\hat{R}_2(h_1) < \hat{R}_2(h_2) \implies \hat{E}_2(h_1) < \hat{E}_2(h_2)$$

Now consider the weight α_{h_1} assigned to ranker h_1 at the first stage, with distribution D_1:

$$\alpha_{h_1} = \frac{1}{2}\log\frac{\epsilon_1^+ + \frac{1}{2}\epsilon_1^0}{\epsilon_1^- + \frac{1}{2}\epsilon_1^0}$$

This equivalency holds because at the first iteration, there is no previous weight α' to take into account.

$$\epsilon_t^+ + \epsilon_t^- + \epsilon_t^0 = \hat{c}$$

Therefore, the linear rank loss function can be represented in terms of epsilon.

Below, rewrite alpha in terms of $\hat{R}_2$.

$$R_2(h_t) = \epsilon_t^- + \frac{1}{2}\epsilon_t^0$$

$$\alpha_{h_1} = \frac{1}{2}\log\left(\frac{\hat{c} - \hat{R}_2}{\hat{R}_2}\right)$$

Assume $\hat{R}_2(h_1) < \frac{\hat{c}}{2}$. (This holds because otherwise we would just use the inversion of the weak ranker, or throw it out if it were exactly $\frac{\hat{c}}{2}$.)

NOTE: Ignore the case where $\hat{R}_2 = 0$ since that would already be a perfect ranker.

Therefore, $\alpha_{h_1} > 0$, meaning the error associated with the weighted weak ranker is the same as its base error, i.e. $\hat{R}_2(\alpha_{h_1} h_1) = \hat{R}(h_1)$. Now, consider the above equations in terms of $\hat{E}_2(\alpha_{h_1} h_1)$.

$$\hat{E}_2(\alpha_{h_1} h_1) = \hat{R}_2(\alpha_{h_1} h_1)(e^{\alpha_{h_1}} - e^{-\alpha_{h_1}}) + \hat{c}e^{-\alpha_{h_1}}$$

Substitute in the values of $\hat{R}_2(\alpha_{h_1} h_1)$ and α_{h_1} in terms of $\hat{R}_2$.

$$
\begin{aligned}
&= \hat{R}_2(h_1)\left(e^{\left(\frac{1}{2}\log\frac{\hat{c}-\hat{R}_2(h_1)}{\hat{R}_2(h_1)}\right)} - e^{-\left(\frac{1}{2}\log\frac{\hat{c}-\hat{R}_2(h_1)}{\hat{R}_2(h_1)}\right)}\right) + \hat{c}\,e^{-\left(\frac{1}{2}\log\frac{\hat{c}-\hat{R}_2(h_1)}{\hat{R}_2(h_1)}\right)} \\
&= \hat{R}_2(h_1)\left(\sqrt{\frac{\hat{c}-\hat{R}_2(h_1)}{\hat{R}_2(h_1)}} - \sqrt{\frac{\hat{R}_2(h_1)}{\hat{c}-\hat{R}_2(h_1)}}\right) + \hat{c}\sqrt{\frac{\hat{R}_2(h_1)}{\hat{c}-\hat{R}_2(h_1)}} \\
&= \hat{R}_2(h_1)\left(\frac{\hat{c}-\hat{R}_2(h_1)}{\sqrt{(\hat{R}_2(h_1))(\hat{c}-\hat{R}_2(h_1))}} - \frac{\hat{R}_2(h_1)}{\sqrt{(\hat{R}_2(h_1))(\hat{c}-\hat{R}_2(h_1))}}\right) \\
&\qquad + \frac{\hat{c}\hat{R}_2(h_1)}{\sqrt{(\hat{R}_2(h_1))(\hat{c}-\hat{R}_2(h_1))}} \\
&= \frac{2\hat{c}\hat{R}_2(h_1) - 2\hat{R}_2(h_1)^2}{\sqrt{(\hat{R}_2(h_1))(\hat{c}-\hat{R}_2(h_1))}} \\
&= \frac{2\hat{R}_2(h_1)(\hat{c}-\hat{R}_2(h_1))}{\sqrt{(\hat{R}_2(h_1))(\hat{c}-\hat{R}_2(h_1))}} \\
&= 2\sqrt{(\hat{R}_2(h_1))(\hat{c}-\hat{R}_2(h_1))}
\end{aligned}
$$

Consider $\hat{R}_2(h_1) < \hat{R}_2(h_2) < \frac{\hat{c}}{2}$.

$$2\sqrt{\hat{R}_2(h_1)(1-\hat{R}_2(h_1))} < 2\sqrt{\hat{R}_2(h_2)(1-\hat{R}_2(h_2))}$$

$$\hat{E}_2(\alpha_{h_1} h_1) < \hat{E}_2(\alpha_{h_2} h_2)$$

□

The introduction of cost does not impact this property of RankBoost+, which maintains this distinguishing feature between the RankBoost versions as the Cost-Sensitive RankBoost algorithm, like standard RankBoost, not be guaranteed to abide by this property.

3.3.2 Generalization Bounds

In order to evaluate the predictive quality of the model, the generalization bounds describe the expected performance difference between the training data and the testing/unknown data. Some of these issues may arise from model instability, meaning that slight variations in the

training data lead to significant differences in the model. Others are due to overfitting where the model is too tightly fixed to its specific training data and therefore fails to generalize to future applications on new data from the same distribution. Freund, et al. proved the generalization bounds for the original RankBoost algorithm using VC-dimension analysis[4], which will be the foundation of the following proofs about the cost-sensitive RankBoost variant introduced above. The proof would be similar for RankBoost+, since the generalization bound considers the expected value of the rank loss and ties are all broken randomly leading to an expected value of 50% correct results from ties. Although the final results will differ slightly between these algorithm variants, their reaction to the incorporation of cost is very similar and therefore one proof is sufficient to demonstrate the effect of creating a cost-sensitive variant on the generalization behavior.

Providing generalization bounds for ranking requires some assumptions, which differ slightly from those necessary for classification. This proof specifically reflects bipartite feedback provided by binary weak rankers, so first assume that there are two underlying distributions of data which are fixed and unknown from which the elements to be ranked are drawn independently. Each pair of elements to be ranked is drawn from this pair of distributions, each of which is i.i.d. Second, restrict the weak rankers to be binary classifiers, which matches existing implementations of the standard and cost-sensitive variants of the RankBoost algorithms and simplifies the proof. To facilitate the proof, consider a basic cost structure with a binary structure of high cost and low cost elements. This will lead to three types of pairs, namely, high-high, low-low, and low-high cost match-ups. However, this clearly generalizes to higher finite cost precision for a more complicated cost model by simply increasing the number of pair categories up to an arbitrary constant. To gain a general understanding about the generalization behavior in the presence of cost, this simplified model is sufficient.

The goal of proving generalization bounds is to ensure that with a high degree of certainty, the difference between the test error and the training error will not exceed some error value ϵ. Essentially, the training error for any model will be sufficiently reflective of the quality of the model on new test data in almost all cases. Freund, et al. showed that the generalization bound for the original RankBoost algorithm is[4]

$$|\hat{\epsilon}(H) - \epsilon(H)| \leq 2\sqrt{\frac{d'(\ln(2m/d')+1)+\ln(18/\delta)}{m}} + 2\sqrt{\frac{d'(\ln(2n/d')+1)+\ln(18/\delta)}{n}}$$

where $d' = 2(d+1)(T+1)\log_2(e(T+1))$ given the weak rankers h_t comprising ensemble ranker H belong to class $\mathscr{H}$ of VC-dimension d. m and n are the sizes of the samples from the two distributions of testing data, and T is the number of weak rankers that are incorporated into the ensemble. This generalization bound states that with probability $1-\delta$ RankBoost-trained rankers will perform within this precision between the training and the testing datasets. This indicates, as matches intuition, that growing the sample size shrinks the generalization error bound and increasing the confidence in the bound by lowering δ naturally increases the range of the bound. Additionally, the number of weak rankers incorporated as denoted by T will also impact the generalization bound though this is mitigated to an extent by the log term. This bound can provide a useful base point to make guarantees with certain levels of confidence about the performance of a model on novel testing data based on its training loss values.

Now, in order to extend this to the cost-sensitive variant of RankBoost described in this thesis, replicate the proof for the original algorithm on the three separate cases described above. This causes the cost values to be a constant that can be factored out of the respective loss equations so that it will essentially emulate binary performance in three separate cases. Recall that this limited proof can be extended to further cost differentiation for some constant number of different pairwise costs by expanding the number of generalization cases considered or by

bounding by the largest cost terms depending on the degree of precision needed for the specific application. This proof will prove a useful starting point for incorporating cost into the generalization bound from the original algorithm. The ensemble ranker, like in standard RankBoost, is constructed as

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where the difference is based on how the optimal weak rankers are selected at each iteration due to optimizing a modified loss function. Since error is calculated pairwise, simply consider the relative position of the two elements in the pair in the ultimate ranking, assigning the pairs values $\{-1, 0, 1\}$ based on this ordering (note that with sufficiently large training sample, ties should occur increasingly rarely between distinct elements as the model becomes more precise). Randomly assign ties and assume they occur with nearly negligible frequency with adequately trained models. From this, the error is computed as

$$\epsilon(H) = Pr_{x\sim D_0, y\sim D_1}[H(x,y) \neq 1]$$

$$= E_{D_0,D_1}[[\![H(x,y) \neq 1]\!]]$$

For proof that this matches the error expressed by the loss function see the original RankBoost paper. The testing error is expressed similarly but is a sum over all pairs normalized by the sampling sizes instead of the expected value, and the expected maximum difference between these values, i.e. $|\epsilon(H) - \hat{\epsilon}(H)|$ is the desired generalization bound. Mathematically, Freund, et al. expressed this objective as

$$Pr_{S_0\sim D_0^m, S_1\sim D_1^n}\left[\exists H \in C : \left|\frac{1}{mn}\sum_{i,j}[[\![H(x_i,y_j) \neq 1]\!]] - E_{x,y}[[[\![H(x,y) \neq 1]\!]]]\right| > \epsilon\right] \leq \delta$$

As mentioned before, in order to extend this proof to cost-sensitive Rankbook it is necessary to consider at least three cases due to differing costs for different pair rankings depending on the relative cost of the elements involved in that pair. Assume without loss of generality that the cost being consider is q, where $q \in$ {low-low, low-high, high-high}. Following along the proof for the

standard RankBoost generalization bound, separately consider the probabilities over sample S_0 and sample S_1, which can each reduce to classification generalization problems. By the principle of triangulation, the net error from both of these components will be bounded by their sum. At this point, the necessary value/properties of ϵ are not clear and are exposed during the proof process. In order to turn the ranking model H into a binary function, consider only pairs with one element drawn from each starting distribution, ignoring $S_0 - S_0$ pairs and $S_1 - S_1$ pairs, and assign each pair a value 0 if it is correctly ordered by H and a value q otherwise using function $F : X \times Y \rightarrow \{0, q\}$. Note: replace the error value 1 with q to keep track of cost in contrast with the original algorithm. Use this indicator function F to rewrite the error difference term described above with the modification to F to accommodate for cost added from the original proof. With each term separated out by its pairwise cost, the error difference term can be rewritten as

$$\frac{1}{mn}\sum_{i,j} F(x_i, y_j) - E_{x,y}[F(x, y)]$$

This can be rewritten by adding and subtracting an identical term, moving expectation and coefficients out, and splitting up summation steps as

$$\frac{1}{m}\sum_{i}\left(\frac{1}{n}\sum_{j} F(x_i, y_j) - E_y[F(x_i, y)]\right) + E_y\left[\frac{1}{m}\sum_{i} F(x_i, y_j) - E_x[F(x, y)]\right]$$

Having these separate terms enables bounding each subproblem with error values ϵ_0 and ϵ_1 respectively, where the two error values sum to the desired maximum error difference ϵ, which when achieved with probability $\geq \frac{\delta}{2}$ each guarantees with high probability that the desired bound will be achieved for a model on a randomly selected test sample from the underlying distributions. This step is slightly complicated by the splitting of the error components based on differing cost values for the pairs, but since the different cost subproblems are disjoint they can be considered separately, and each component can similarly be reduced to its respective share of the model error on testing data, which will then symmetrically transpose to the generalization for classification as used by Freund, et al. From there, consider the $\bigcup \mathscr{F}_y$ where $\mathscr{F}_y$ describes the relationship of each y value to all x values. Using VC-dimension properties from

Vapnik[50], for any $\delta > 0$,

$$Pr_{S_0 \sim D_0^m}\left[\exists F \in \bigcup \mathscr{F}_y : \left|\frac{1}{m}\sum_i F(x_i, y) - E_x[F(x, y)]\right| > \epsilon_0(m, \delta, d')\right] < \delta$$

where d' is the complexity of the different $\mathscr{F}_y$ functions and

$$\epsilon_0(m, \delta, d') = 2\sqrt{\frac{d'(\ln(2m/d') + 1) + \ln(9/\delta)}{m}}$$

Since the sample size m and the probability bound δ are both known, only d', the VC-dimension of the y-functions, must be calculated. Start by expanding the generic binary error mapping function F as laid out below.

$$F(x, y) = q\left[\!\!\left[\sum_{t=1}^{T} \alpha_t h_t(x) - b \geq 0\right]\!\!\right]$$

where q is the cost being considered by the current subproblem and $b = \sum_{t=1}^{T} \alpha_t h_t(y)$ is fixed for each $\mathscr{F}_y$. Earlier work by Freund and Schapire[46] bounds this error in terms of the VC-dimension $d \geq 2$ of the distribution of weak rankers $\mathscr{H}$ where

$$d' \leq 2q(d+1)(T+1)\log_2(e(T+1))$$

These differing values based on the q cost value for each d' will henceforth be notated as d'_q for a given cost value. Following along the proof for RankBoost's generalization bound and performing the symmetrical operation for all values of x being considered by the ranker, gives a final generalization bound of

$$|\hat{\epsilon}(H) - \epsilon(H)| \leq \sum_{q=1}^{3}\left(2\sqrt{\frac{d'_q(\ln(2m_q/d'_q) + 1) + \ln(18/\delta_q)}{m_q}} + 2\sqrt{\frac{d'_q(\ln(2n_q/d'_q) + 1) + \ln(18/\delta_q)}{n_q}}\right)$$

for this cost-sensitive RankBoost variant, where m_q and n_q denote the disjoint subsets of those samples which are assigned these cost values and δ_q is normalized to account for the proportion of pairs which are assigned cost q. Although similar to the generalization bound determined by Freund, et al. for the original version of the algorithm, the pairwise costs are

featured in the bound, emphasizing the performance differences due to the addition of cost-sensitivity to the algorithm. This indicates that for a given dataset drawn from some two distributions, models which have the same cost-blind performance will have tighter error generalization bounds given relative emphasis placed on correctly ranking higher cost pairs.

3.3.3 Other Boosting Algorithms

Using a pairwise cost metric with options for discrete or continuous value assignment is easily extensible to other boosting algorithms, whether they are to be used for ranking or another application. Introducing a cost in a way that intuitively emphasizes the relative importance of correctly training a model to characterize certain vital elements is a simple procedure. However, it is important to ensure that the cost is consistently represented in the loss function optimized to train the specific model. Furthermore, this cost sensitive paradigm could also be extended to listwise boosting algorithms such as Adarank[7]. Either pariwise or individual costs would be appropriate for such algorithms, and these costs could similarly be introduced at any stage of the training process (i.e. the initial distribution, per training iteration, or as a post-processing alteration). The vital property is that cost should be separated from the label or rank that is being assigned, since cost should represent non-redundant information about which elements deserve prioritization and will need special emphasis during training to ensure their correct treatment by the model.

4 Experiments

The experiments for this thesis use Rankboost, Rankboost+, and their cost-sensitive variants coded in Python 3.7. The weak learners are decision stumps, and the tests are run with 5-fold cross validation. Due to concerns with computability, full linear independence is not proven for the weak rankers chosen by Rankboost+ and cost-sensitive Rankboost+. However, heuristics to reduce redundancy, which in the limit results in linear independence, approximate this property[11]. First, this section will introduce the datasets used and the metrics which allow performance comparisons between the different algorithm variants. Then, all of the experiments are described, leading to some general observations and integrative analysis on how the different focus areas of the experiments reflect the overall properties of the algorithms.

4.1 Cost-Sensitive Datasets

Although many ranking datasets are publicly available, there are not labeled cost-sensitive datasets for this problem yet. However, there are a number of different experimental approaches described here which can provide insight into the performance differences elicited by applying cost to Rankboost and Rankboost+. Some ranking datasets have a built-in sense of "cost," such as the salary of football players. In that example, the cost will already be correlated with the ultimate ranking though, and therefore will not give a position agnostic sense of importance that characterizes the general version of the problem. Based on the Rankboost+ paper[11], which used the MovieLens dataset[51], the first experiment is a modification of that ranking problem

with randomized costs assigned in two trials, with a binary individual cost structure (low vs. high) and a smoother random distribution of costs. Second, for the experiment for multiclass cost-sensitive classification uses the UCI LIBRAS dataset based on the work by Beijbom, et al[13]. To compare cost-sensitivity to bias, the third experiment uses the COMPAS dataset for recidivism released by Pro Publica which elicited so much outcry about reducing bias in ML[22]. This experiment was based on the work done by other researchers working in reducing ranking bias for ML who also used the COMPAS dataset[34–36]. Each of these experiments investigates a different aspect of the cost-sensitive ranking problem, and by close analysis of the results it is possible to establish how the performance is distinguished between the original and cost-sensitive versions of the algorithms.

Datasets and their Properties			
Dataset	MovieLens	LIBRAS	COMPAS
Cost	Random	SVM-like	Bias
Ranking	Yes	Multiclass	No
Features	1128	91	11
Classes	n	15	2

Table 4.1. Overview of Experimental Datasets with Properties

4.2 Performance Metrics for Cost-Sensitive Ranking

In order to compare the performance of the different variants, particularly between the cost-sensitive and original versions of the algorithms, outside metrics are necessary. However, there are no extant cost-sensitive ranking metrics available. Some suggestions follow, and discussions of their properties and shortcomings will inform their usage to compare the experimental results across algorithms. However, these metrics will not be extensively evaluated in this paper, and determining the properties of cost-sensitive ranking metrics would be an interesting avenue for future work.

4.2.1 Rank Loss

In some ranking problems, there may be a tradeoff between uniform rank loss and cost-sensitive rank loss. However, this is not necessarily the case, as the cost-sensitive algorithm ideally will still achieve comparable performance across all pairs, but with greater emphasis on correctly ranking certain "high-cost" pairs. If multiple models with the same performance quality are available, the cost-sensitive variant of an algorithm will train a model which prioritizes certain pairs based on their relative importance, whereas the standard version will not distinguish between the models during training in this way since all pairs will be considered equally important. With complete "oracle" optimization to train an entire model, for the original Rankboost algorithms, the uniform rank loss should never be lower than the uniform rank loss achieved by their respective cost-sensitive variants. The opposite should be true for cost-sensitive rank loss, where cost-sensitive variants of the algorithms should perform at least as well as the original versions. In general, this should apply empirically; however, these suppositions may fail in certain edge cases due to the greedy, iterative training of these boosting algorithms. For the purposes of this paper, observing the differences between rank loss and cost-sensitive rank loss performance between the different algorithm variants to be tested will be useful to develop intuition about performance. Computing more rigorous mathematical comparisons between cost-sensitive performance and uniform cost performance of different algorithms is a potential area for future research.

4.2.2 NDCG

One of the most common metrics applied to evaluate the efficacy of a ranker is Discounted Cumulative Gain (DCG), which is often normalized (NDCG). NDCG works by weighting the distance from the correct rank according to the true rank, so there is a higher penalty for mistakes in the top of the rank[10]. This approach reflects the ethos of information retrieval, where

only the ranks at the top matter. Many different discounting factors have been tried, including "top-K" approaches and the Zipfan discount (r^{-1}), but the generally accepted standard is to use $\log(1+r)^{-1}$, logarithmic decay[10]. Here, D is the choice of discounting factor, $D(r)$ is the discounted rank of the current element, f is the ranking function, and y is the magnitude of the difference of rank assigned to the element by the model and its true rank. The denominator is a normalizing factor computed as the optimal value of NDCG for any ranker on the given dataset, the Ideal Discounted Cumulative Gain (IDCG).

$$NDCG_D(f, S_n) = \frac{\sum_{r=1}^{n} y_{(r)}^{f} D(r)}{\max_{f'} \sum_{r=1}^{n} y_{(r)}^{f} D(r)}$$

Very limited work has been done evaluating the theoretical properties of NDCG, but its widespread usage makes it an important metric to consider when working with ranking. According to Wang, et al., NDCG does achieve some provable guarantees in the limit, which seems to be the most extensive work evaluating the metric's properties[10]. Their paper introduces "consistent distinguishability," meaning that with sufficiently large testing samples from the same distribution, the better performing model will be "distinguished" as having a better NDCG value. An interesting point introduced by this research is that this property does not hold for all discount factors, with r^{-1} being the limit of rapid discounting where the NDCG values result in consistent distinguishability. Additionally, it only works for continuous evaluation, so the top-K models also do not qualify. The conclusion reached by Wang, et al. is that while a range of discounting factors are valid according to this property, the logarithmic discount is validated as a mathematically robust choice.

Unfortunately, using basic NDCG does not extend well to the problem of cost-sensitive ranking since the discounting contradicts the notion of position-agnostic cost functions. The point of having high cost on pairs that include lower ranked elements is to prioritize correctly ranking

those pairs, so if they are less relevant to the metric that will not sufficiently reflect the performance of the cost-sensitive model. This thesis proposes the following variation which accommodates for cost-sensitivity, although its non-continuity means it will not necessarily fit the property of consistent distinguishability explained above.

$$CSDCG_p = \sum_{j=1}^{p} \frac{2^{rel_j} - 1}{\log_2(i+1)} \eta_1(j) - \frac{2^{rel_j} - 1}{\log_2(i+1)} \eta_2(j)$$

$$\eta_1(j) = \begin{cases} c_j^* & \text{if j is supposed to be in the top half of all results} \\ 0 & \text{otherwise} \end{cases}$$

$$\eta_2(j) = \begin{cases} 0 & \text{if j is supposed to be in the top half of all results} \\ c_j^* & \text{otherwise} \end{cases}$$

$$c_j^* = \frac{1}{m-1} \sum_{j=1}^{m} \tilde{c}(i,j); \qquad \tilde{c}(x,x) = 0$$

In order to retain the importance of high-cost but low ranked elements, this modified DCG separately evaluates results in the top half and the bottom half of the true rank. Each result is weighted by its cost factor, which is computed using the average of its pairwise costs. The score associated with items in the bottom half of the rank is subtracted, so correctly scoring important low-rank, high-cost elements still has a significant impact on the score. The top part of the rank is similar, so there will be greater penalties for misranking a high-cost element than a lower cost one. This still uses discounting rather than considering every rank position equally, so with normalization it will be comparable to standard NDCG. However, an extreme cost structure can severely impact this score, so it will be more difficult to compare between datasets with significantly differing cost assignments. As a rule-of-thumb, the cost values should be within

an order of magnitude of each other in order to prevent heavyweight elements from dominating the CSDCG result. Without a clearer imperative to use this metric, it too serves to provide insight into behavior without strictly proving comparative properties between the algorithm variants for each experiment.

4.3 Experimental Results

In this section, several different experiments are described to illustrate and explore various properties of the cost-sensitive Rankboost variants introduced in this paper. Although there are not canonical ranking datasets with cost-sensitivity, interesting insights arise from observing a variety of relevant ranking and cost-sensitive scenarios, especially contrasting the specific behavior of the cost-sensitive variants compared to the original Rankboost algorithms on elements and pairs where cost plays a role.

4.3.1 Randomized Cost

In order to directly compare the performance of the cost-sensitive versions of Rankboost and Rankboost+, this experiment uses the MovieLens dataset and follows a similar experimental procedure to the RankBoost+ paper[11,51,52]. Since the MovieLens dataset is not cost-sensitive, the comparison is by nature somewhat contrived, but it provides valuable insight into the behavior of the cost-sensitive variants compared to the standard approach. Two approaches to randomizing costs were applied in separate experiments to analyze the impact on performance. In one, shown in Table 4.2 as 3-MovieLens, ternary weights were assigned to each element with values $1\frac{1}{2}$, 1, and $\frac{1}{2}$ (see Figure 4.2). The second experiment assigned real values within the range 0 to 2 to each element (see Figure 4.1). The MovieLens dataset provides a dense feature matrics with 1128 features given as affinity probabilities; these features condense reviews into a score and a wide-range of tags which can indicate movie theme and potentially correlate with audience enthusiasm. This experiment used 1000 of those features and randomly sampled from the

corpus of movies for training and testing sets.

The Cost-Sensitive Rankboost+ performance was similar to Rankboost+ in both the cost-sensitive

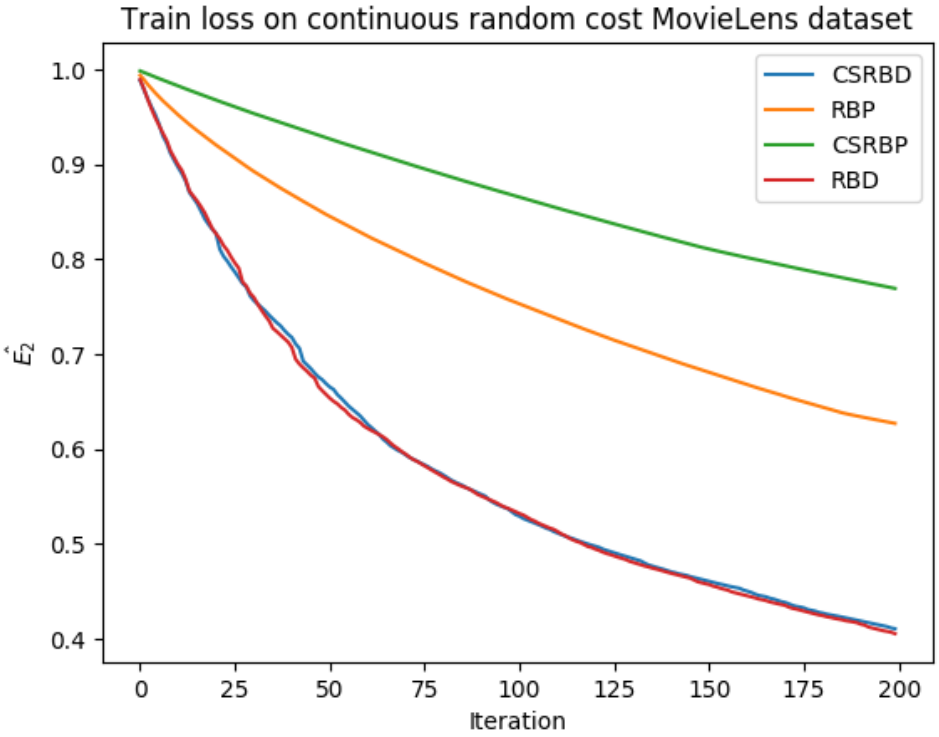

Figure 4.1. Rank Loss Convergence During Training with Continuous Costs

and standard exponential and rank loss metrics, but performed slightly worse in NDCG (see Table 4.2). A similar pattern was evident for Cost-Sensitive Rankboost, but to a lesser extent due to the overfitting displayed by both versions (which led to lower rank loss akin to memorization on the training data). Accordingly, there appeared to be a slight improvement in the generalization of Rankboost+ compared to Rankboost for both the original and cost-sensitive versions, respectively. The normalized CSDCG results for each algorithm are included for completeness, although the lack of implicit cost in the MovieLens dataset makes this value difficult to meaningfully interpret. Because the cost for the testing dataset does not reflect specific properties that align with the training dataset, the cost does not imply true importance differences between the pairs as it would for a canonically cost-sensitive ranking dataset, which would make the CSDCG results more interesting to analyze. Running Rankboost+ on the cost-sensitive dataset,

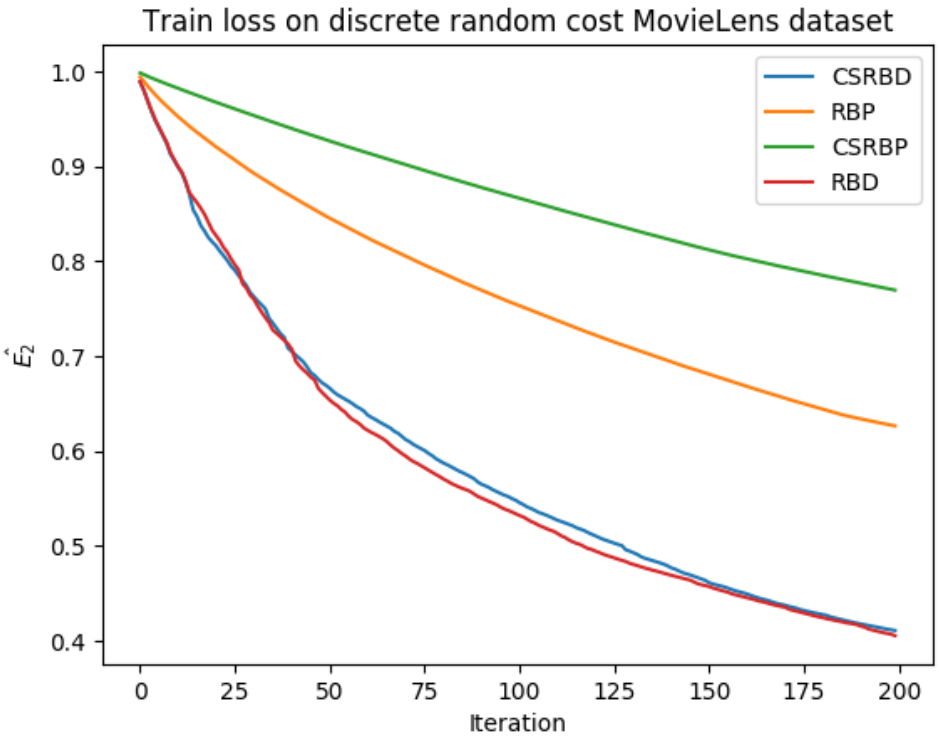

Figure 4.2. Rank Loss Convergence During Training with Discrete Costs

this cost-sensitive version had a marginal advantage in the cost-sensitive values of the metrics with the discrete cost assignments, Figure 4.2, but actually performed worse on the continuous cost assignments, Figure 4.1. This result is likely due to some pairs receiving too low a cost to be considered in time to correctly rank them as compared to the cost-blind version of the algorithm. These differences in performance coincide with reasonable expectations, since the cost-sensitivity is focusing on the performance of the rankers on high-priority, costly elements, which is not considered in the basic version of the algorithm.

A qualitative analysis of the ranking results from the basic and cost-sensitive versions of the algorithm reinforced the intuition about prioritization described above. Cost-sensitive RankBoost+ performed slightly worse on low cost elements, about the same on average cost elements, and slightly better on high-cost elements. Over 10 trials of the experiment seeded with different random values, the ensemble ranker trained by RankBoost+ correctly ranked 84.3%

of high cost pairs, 87.1% of medium cost pairs, and 83.1% of low cost pairs, while on the same training data the ensemble trained by RankBoost+ correctly ranked 88.0% of high cost pairs, 84.4% of medium cost pairs, and 79.2% of low cost pairs. This performance difference appeared consistent with respect to position in the final ranking, so this investigation seems to support the goal of achieving similar performance to cost-blind ranking algorithms, while directing the focus to specific high priority elements. Because cost was randomly assigned, improvements in standard performance metrics would be anomalous, as the costs do not in this case reflect a meaningful subset of elements along the margin to direct the attention of the algorithm like an SVM. It is interesting to note that randomly assigned continuous costs detracted from the performance of the cost-sensitive versions of the algorithm, though, since this indicates a potential challenge of using this model if cost becomes too noisy.

Empirical Performance of Each Algorithm				
Dataset and Metric	Rankboost	Rankboost+	Cost-Sensitive RBD	Cost-Sensitive RB+
3-MovieLens $\hat{R}_2$	0.443	0.412	0.467	0.398
3-MovieLens NDCG	0.75	0.78	0.71	0.78
3-MovieLens CSDCG	0.65	0.70	0.70	0.68
Multiclass $\hat{R}_2$	0.079	0.084	0.071	0.082
Multiclass Accuracy	0.87	0.93	0.88	0.93
COMPAS $\hat{R}_2$	0.044	0.054	0.039	0.063
COMPAS Accuracy	0.68	0.67	0.69	0.68

Table 4.2. Summary of Empirical Results on 5-Fold Cross Validation Tests for Each Cost-Sensitive Experiment (MovieLens results come from discrete cost implementation, and reported CSDCG is normalized)
Accuracy is the unweighted proportion of correctly labeled elements.

4.3.2 Ranking to Multiclassify with Cost

Cost-sensitivity can provide important insight into margin enforcement for multiclass classification algorithms. Though multiclass classification is not an equivalent problem to ranking,

it is interesting to see how the cost-sensitive ranking algorithm versions perform on the cost-sensitive multiclass classification datasets. In this case, assigning costs to pairs instead of computing it as a function of individual costs provides the most utility. Pairs of similar elements that need to be classified differently are assigned the most weight, analogous to the support vector elements of an SVM along the margin. This experiment involved running the cost-sensitive versions of RankBoost and RankBoost+ on the dataset and comparing the performance both to the performance of the standard RankBoost algorithms and the cost-sensitive multiclassification results achieved by Beijbom et al[13]. The dataset used was UCI LIBRAS (Brazilian Sign Language), which has 15 classes and 91 attributes. Cost was assigned by recreating the process described in Beijbom, et al. which assigns symmetric, real-valued costs pairwise between each of the different classifications[13].

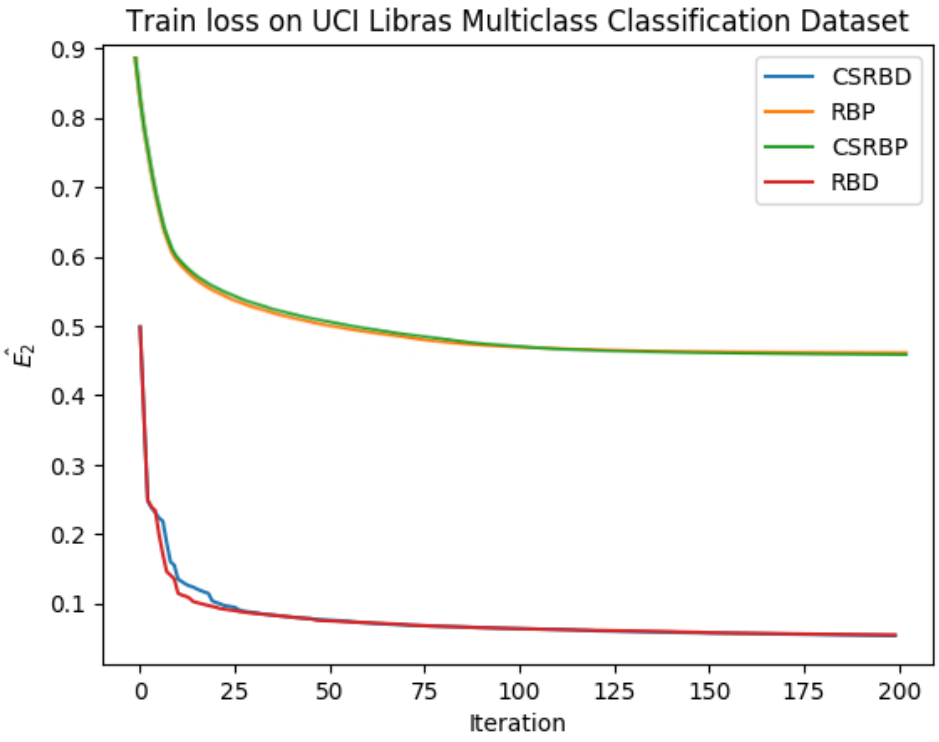

Figure 4.3. Rank Loss Convergence During Training for Multiclass Classification

Interestingly, although the $\hat{E}_2$ values were even more significantly different here than in the previous experiment as seen in Figure 4.3, the performance on the testing data was similar (see Table 4.2). For both RankBoost and RankBoost+ the cost-sensitive variants had slight deviations from the standard version in terms of exponential loss, but converged to the same value. However, in contrast to the result from the previous experiment, the RankBoost and cost-sensitive RankBoost algorithms outperformed the RankBoost+ and cost-sensitive RankBoost+ algorithms by a consistent and notable amount (though both variations performed fairly well on this multiclass classification task). However, this difference in performance is not directly related to the high disparity between the $\hat{E}_2$ values. Instead, the pattern implies that the requirement in RankBoost+ for linear independence (low-redundancy) between weak rankers does not improve performance for multiclass classification tasks. Whereas in the ranking experiment on the MovieLens dataset above, RankBoost appeared to overfit to the training data, this behavior does not appear in separating the 15 LIBRAS classes. Perhaps because this class is more of a classification task than a full ranking task, the interventions to prevent ties in RankBoost+, and the requirement of linear independence, actually slightly depress performance. Here, ties are an important and necessary part of the ranker's job in order to group elements into the correct class label, so increasing the focus and penalty on ties loses utility compared to the ranking task. Although the results are not conclusive in this area, it is also worth noting that both cost-sensitive variants outperformed their parent algorithms in the rank loss for the testing data. Despite this experiment not being a true ranking problem, it provides more insight into the performance of these cost-sensitive algorithm variants on a cost-sensitive application and builds understanding about the range of capabilities and limitations for all four implementations tested.

4.3.3 COMPAS

The COMPAS recidivism dataset is not just a symbolic justification for the focus on bias-free and cost-sensitive machine learning. While it is useful to highlight the potential shortcomings of using ML for sensitive applications such as determining which prisoners get parole, it also has several interesting properties which make it relevant to work with cost-sensitive ML. In particular, the class asymmetry between the representation of black and white prisoners can be transposed into a cost-sensitive problem by shifting the threshold for marginalized demographics of prisoners who are unfairly overrespresented in the dataset. These initial forays into using cost-sensitive Rankboost for this task are by no means sophisticated enough to achieve the bias mitigation objectives discussed in Chapter 2, but nonetheless provide some insight into how cost-sensitivity can be used to shift a model to cost-sensitive version with similar performance. Since COMPAS is a classification dataset, the ranking has little real-world capital since it does not even correspond to a calibration probability of recidivism. However, the values spit out provide a preferential ordering of inmates for parole which can reflect biases in the dataset and help to explain what factors correctly and erroneously (from a social perspective) influence the outcome of the model training.

This experiment was based on the choices made by Asudeh, et al. for what features to use and how to evaluate the output[35]. Accordingly, this model was trained on the output from COMPAS, timing of the arrest, the count of non-felony juvenile offenses, the total count of prior offenses, and demographic information such as sex, age, and race. All the categorical variables were encoded as integers. Because ProPublica provided a "fair" outcome for this data, that served as the correct label for each element. In order to discern the impact of cost on the fairness of the outcome, the cost structure placed more weight on training pairs with the protected class; in this case, the protected class was non-white prisoners. Although the behavior varied slightly

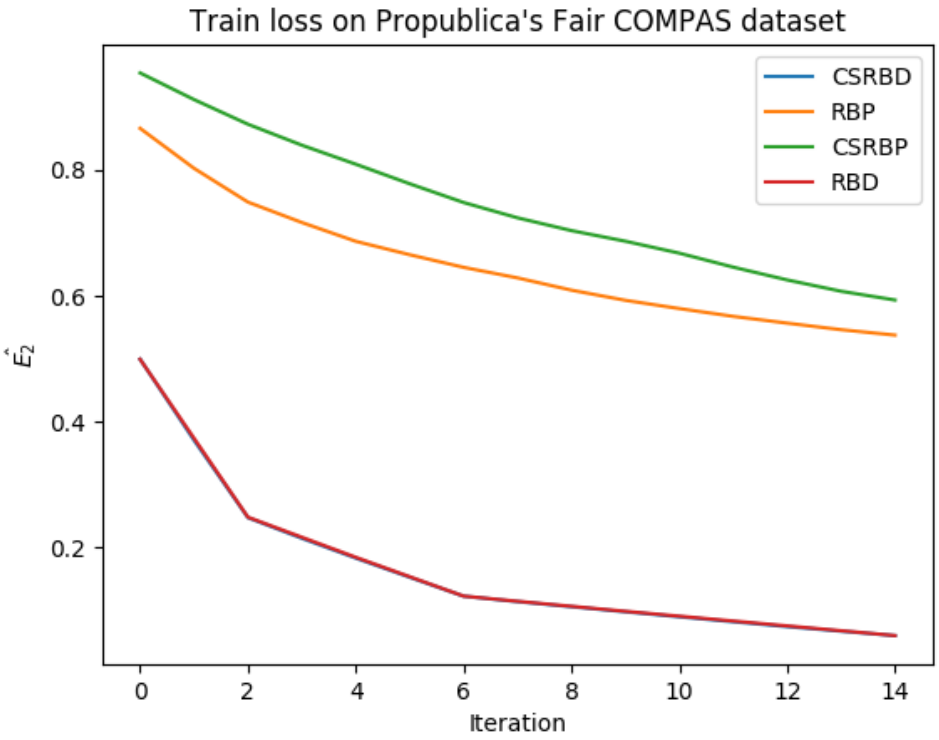

Figure 4.4. Rank Loss Convergence During Training for Recidivism Classification

between the cost-sensitive and original variants of the algorithms, due to the binary classification nature of the problem and the difference in objective from fair ranking experimentation on this dataset, making broad conclusions is difficult. For example, Asudeh, et al. ran thousands of experimental iterations and used statistical fairness metrics to determine which proportion produced a fair classification outcome with regard to the protected class[35]. Overall, the performance, and the models themselves, were very similar due to the limited number of features available (see Figure 4.4 and Table 4.2). That said, a qualitative assessment across several iterations of this experiment indicated some promise to this approach, since the models constructed with cost appeared to select rankers which placed a stronger emphasis on correctly ranking the protected class. However, in the absence of a fair ranking metric or a dataset with more than binary separation, these speculations remain an important area for future, more systematic investigation.

4.3.4 Performance

Overall, the experiments were not the central focus of this research, and instead should be taken as explorations of different aspects of this work in cost-sensitive ranking. In particular, the third experiment on the COMPAS dataset was intended to investigate the work done by other researchers into fair ranking rather than to seriously propose a robust fair solution to that problem using cost-sensitive ranking. Although the problems of fair ranking and cost-sensitive ranking are closely intertwined, without a more structured interplay to incorporate cost in a way that promotes fairness (by some metric), cost-sensitive algorithms cannot solve the problem of bias in machine learning on their own.

Although the dataset was not specifically intended for cost-sensitivity, the most interesting experiment from a ranking perspective was the MovieLens random-cost one. Because the final ranking was fully stratified, unlike the other two experiments which used ranking for classification, this experiment elicited the most significant performance difference during training between the algorithms. Here, cost played the most significant role in changing the models that were trained by each implementation, even between cost-sensitive and standard RankBoost. Overall, the cost-sensitive variants throughout these three experiments achieved similar performance as their base algorithms (see Table 4.2), but trained different models specifically due to the changes in cost. This section therefore provides a basic proof-of-concept that this cost-sensitive approach is a workable way to incorporate cost into ML ranking algorithms.

Due to the shortcomings of the datasets in terms of having well-formulated costs assigned to

each pair, it is difficult to extensively assess the differences in performance between the cost-sensitive versions of RankBoost and RankBoost+ versus their parent algorithms. However, because the performance remained similar between each algorithm and its cost-sensitive derivative, these tests empirically support a significant point upon qualitative examination. Particularly in the MovieLens tests, since those actually elicited a full ranking, the cost-sensitive variants prioritized correctly ordering high cost pairs as compared to the standard versions of the algorithms. These changes resulted in very similar NDCG and RankLoss scores during testing, which seems to indicate that the cost-sensitive variants found similarly performing models which achieved their cost-sensitive objectives. This result is promising for the application of cost-sensitive ranking algorithms to fair ranking problems, since it shows that the algorithms can be adjusted to asymmetric datasets.

5 Discussion

This chapter wraps up the thesis with a discussion of potential areas for future work into the topics herein, and a conclusion which summarizes the contributions made to this research area.

5.1 Future Work

Machine learning fairness is attracting a lot of research attention right now, and will likely continue to grow for some time. This is positive in that the amount of interest and effort will lead to higher standards of quality and better solutions, but may also lead to confusion and fragmentation if no clear set of solutions is able to dominate common practice. There is also likely to be more work done on individual and subgroup fairness due to the relative stability of the metrics used to evaluate group fairness, and their situational shortcomings. A related issue is fairness for deep learning, which right now is concentrating heavily on explainability for neural networks. There is still a long way to go before clear theory can be introduced to guarantee fairness for neural networks, though the work on evaluating the underlying semantics of a network's decisions is an important step of the process.

A driving external factor which is most likely to majorly impact the work in fair ML is legal regulation. Although technology is famously underregulated due to its quick evolution and seeming impunity from legal responsibility, as ML begins to play a role in increasingly sensitive applications, eventually regulation must try to catch up. Legal interventions requiring explainability

and even fair treatment according to some metric will drive development in this field and also bring a much higher degree of outside scrutiny. Having theoretically robust fair algorithms will play an important role in this eventuality, especially due to the widespread perception that algorithms are racist, sexist, etc. which will lead to pushback against their spreading usage for official or otherwise important functions. If legal regulations cement certain approaches as the best for sensitive applications, ideally those choices will reflect the state-of-the-art and will actually promote fairness or detect unfairness in a way that is consistent, explainable, and efficient.

One obvious avenue for continued research is figuring out how to extend fairness and cost-sensitivity to non-boosting ranking algorithms, or even variants that differ from the Rankboost algorithms explored in this paper. Another important area of future theory work for cost-sensitive ranking is proving performance bounds on the metrics used to compare and evaluate the different algorithms' experimental behavior. There is also room for new cost-sensitive ranking metrics or improvements to existing ones, for example modifying NDCG to evaluate cost-sensitive performance in a way that is less brittle. Ultimately, one of the biggest open questions from this paper is how to integrate the work on cost-sensitivity into fair ranking in a theoretically sound way. The work done in classification and the inherent connections between the problem spaces provide an interesting grounds for further research.

5.2 Conclusion

This thesis provided several important theoretical results that expand the understanding of the fair and cost-sensitive ranking problems. The first chapter provided a literature review on an overview of the ranking problem, efforts towards fair ML and reducing bias, and class asymmetry including cost. The second chapter introduced new standards by which to evaluate bias mitigation strategies for ranking and demonstrated the shortcomings of the metrics currently in

use. In particular, metrics that suffice for fair classification problems fail to capture the nuance of the full ordering required for ranking. As a result, many metrics are susceptible to fundamental challenges such as detecting tokenization, which can deeply undermine the success of attempts to create fair models. Although there has been some recent work in this area, the literature lacks a robust investigation of the additional complications in fair ranking over fair classification.

The third chapter introduces a position-agnostic cost mechanism to indicate objective importance of pairs for boosted ranking algorithms, and provided proofs of several properties of this approach to cost including generalization bounds. As alluded to in the two previous chapters, having cost-sensitive algorithms can aid in the development of fair models through a reductions approach[29] or by improving individual fairness outcomes and model precision on a preprocessed or audited dataset. The fourth chapter provides the accompanying cost-sensitive experimental results with analysis and a brief analysis of metrics for cost-sensitive ranking. Overall, this thesis provides a foundation to approach fair and cost-sensitive ranking problems from a robust theoretical perspective, challenges the current usage of classification approaches with insufficient modification to the fair ranking problem, and introduces a general cost mechanism for boosted ranking algorithms. With the many sensitive applications for ML ranking, it is imperative to get the theory right from the beginning.

Complete References

[1] Nikolaos Nikolaou. Cost-sensitive boosting: a unified approach. 2016.

[2] Hang LI. A short introduction to learning to rank. IEICE Transactions on Information and Systems, E94.D(10):1854–1862, 2011.

[3] Serge Abiteboul, Mihai Preda, and Gregory Cobena. Adaptive on-line page importance computation. In Proceedings of the 12th International Conference on World Wide Web, WWW '03, page 280–290, New York, NY, USA, 2003. Association for Computing Machinery.

[4] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res., 4:933–969, December 2003.

[5] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, page 2221–2231, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Carlos Castillo. Fairness and transparency in ranking. SIGIR Forum, 52(2):64–71, January 2019.

[7] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, pages 391–398, 07 2007.

[8] Bita Shams and Saman Haratizadeh. Graph-based collaborative ranking. CoRR, abs/1604.03147, 2016.

[9] Chris J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft, June 2010.

[10] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Tie-Yan Liu, and Wei Chen. A Theoretical Analysis of NDCG Type Ranking Measures. arXiv e-prints, page arXiv:1304.6480, Apr 2013.

[11] Harold Connamacher, Nikil Pancha, Rui Liu, and Soumya Ray. Rankboost+: an improvement to rankboost. Machine Learning, Aug 2019.

[12] Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'07, page 897–904, Red Hook, NY, USA, 2007. Curran Associates Inc.

[13] Oscar Beijbom, Mohammad Saberian, David Kriegman, and Nuno Vasconcelos. Guess-averse loss functions for cost-sensitive multiclass boosting. In Eric P. Xing and Tony Jebara, editors, Proceedings of the 31st International Conference on Machine Learning, volume 32

of Proceedings of Machine Learning Research, pages 586–594, Bejing, China, 22–24 Jun 2014. PMLR.

[14] Bernardo Ávila Pires, Mohammad Ghavamzadeh, and Csaba Szepesvári. Cost-sensitive multiclass classification risk bounds. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, page III–1391–III–1399. JMLR.org, 2013.

[15] Po-Lung Chen and Hsuan-Tien Lin. Active learning for multiclass cost-sensitive classification using probabilistic models. In Proceedings of the 2013 Conference on Technologies and Applications of Artificial Intelligence, TAAI '13, page 13–18, USA, 2013. IEEE Computer Society.

[16] Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. In Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI'06, page 567–572. AAAI Press, 2006.

[17] Ron Appel, Xavier Burgos-Artizzu, and Pietro Perona. Improved multi-class cost-sensitive boosting via estimation of the minimum-risk class. arXiv e-prints, 07 2016.

[18] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Transactions on Neural Networks and Learning Systems, 29(8):3573–3587, Aug 2018.

[19] Muhammad Bilal Zafar, Isabel Valera, Manuel Rodriguez, Krishna Gummadi, and Adrian Weller. From parity to preference-based notions of fairness in classification. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 229–239. Curran Associates, Inc., 2017.

[20] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. A comparative study of fairness-enhancing interventions in machine learning. In Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* '19, page 329–338, New York, NY, USA, 2019. Association for Computing Machinery.

[21] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS '12, 2012.

[22] Surya Mattu Julia Angwin, Jeff Larson and Lauren Kirchner. Machine bias. ProPublica, 2016.

[23] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In Proceedings of the 26th International Conference on World Wide Web, WWW '17, pages 1171–1180, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[24] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big Data, 5(2):153–163, Jun 2017.

[25] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Learning fair classifiers. In Proceedings of Machine Learning Research, 2015.

[26] Claudia Goldin and Cecilia Rouse. Orchestrating impartiality: The impact of "blind" auditions on female musicians. American Economic Review, 90(4):715–741, September 2000.

[27] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, pages 259–268, New York, NY, USA, 2015. ACM.

[28] William Dieterich, Christina Mendoza, and Tim Brennan. Compas risk scales : Demonstrating accuracy equity and predictive parity performance of the compas risk scales in broward county. Northpointe Inc. Research Department, 2016.

[29] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. A reductions approach to fair classification. In FATML'17. Association for Computing Machinery, March 2018.

[30] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness, 2017.

[31] Caitlin Kuhlman, MaryAnn VanValkenburg, and Elke Rundensteiner. Fare: Diagnostics for fair ranking using pairwise error metrics. In The World Wide Web Conference, WWW '19, page 2936–2942, New York, NY, USA, 2019. Association for Computing Machinery.

[32] Michael P. Kim, Amirata Ghorbani, and James Zou. Multiaccuracy: Black-box post-processing for fairness in classification. In Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES '19, pages 247–254, New York, NY, USA, 2019. ACM.

[33] Michael Kearns, Aaron Roth, and Saeed Sharifi-Malvajerdi. Average individual fairness: Algorithms, generalization and experiments, 2019.

[34] Yifan Guan, Abolfazl Asudeh, Pranav Mayuram, H. V. Jagadish, Julia Stoyanovich, Gerome Miklau, and Gautam Das. Mithraranking: A system for responsible ranking design. In Proceedings of the 2019 International Conference on Management of Data, SIGMOD '19, page 1913–1916, New York, NY, USA, 2019. Association for Computing Machinery.

[35] Abolfazl Asudeh, H. V. Jagadish, Julia Stoyanovich, and Gautam Das. Designing fair ranking schemes. In Proceedings of the 2019 International Conference on Management of Data, SIGMOD '19, page 1259–1276, New York, NY, USA, 2019. Association for Computing Machinery.

[36] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa*ir. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17, 2017.

[37] Le Chen, Ruijun Ma, Anikó Hannák, and Christo Wilson. Investigating the impact of gender on rank in resume search engines. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18, New York, NY, USA, 2018. Association for Computing Machinery.

[38] Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In Proceedings of the 29th International Conference on Scientific and Statistical Database Management, SSDBM '17, New York, NY, USA, 2017. Association for Computing Machinery.

[39] A. Wald and J. Wolfowitz. On a test whether two samples are from the same population. Ann. Math. Statist., 11(2):147–162, 06 1940.

[40] Department of Labor Equal Employment Opportunity Commission. Part 1607—uniform guidelines on employee selection procedures, 1978.

[41] Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification. In Lecture Notes in Computer Science, volume 2533, pages 365–379, 11 2002.

[42] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. J. Mach. Learn. Res., 1:113–141, September 2001.

[43] Feiyang Pan, Xiang Ao, Pingzhong Tang, Min Lu, Dapeng Liu, and Qing He. Towards reliable and fair probabilistic predictions: field-aware calibration with neural networks. CoRR, abs/1905.10713, 2019.

[44] Anna Nguyen, Tobias Weller, and York Sure-Vetter. Making neural networks FAIR. CoRR, abs/1907.11569, 2019.

[45] P. Manisha and Sujit Gujar. A neural network framework for fair classifier. CoRR, abs/1811.00247, 2018.

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

[47] Paul A. Viola and Michael J. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In NIPS, 2001.

[48] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci., 55(1):119–139, August 1997.

[49] Liwei Wang, Masashi Sugiyama, Zhaoxiang Jing, Cheng Yang, Zhi-Hua Zhou, and Jufu Feng. A refined margin analysis for boosting algorithms via equilibrium margin. J. Mach. Learn. Res., 12(null):1835–1863, July 2011.

[50] Vladimir Vapnik and S. Kotz. Estimation of Dependences Based on Empirical Data: Empirical Inference Science (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg, 2006.

[51] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst., 5(4), December 2015.

[52] Jesse Vig, Shilad Sen, and John Riedl. The tag genome: Encoding community knowledge to support novel interaction. ACM Trans. Interact. Intell. Syst., 2(3), September 2012.