Reduction from Cost-sensitive Ordinal Ranking to Weighted Binary Classification

Hsuan-Tien Lin[1] and **Ling Li**[2]

[1]Department of Computer Science, National Taiwan University.

[2]Department of Computer Science, California Institute of Technology.

Keywords: cost-sensitive, ordinal ranking, binary classification, reduction

Abstract

We present a reduction framework from ordinal ranking to binary classification. The framework consists of three steps: extracting extended examples from the original examples, learning a binary classifier on the extended examples with any binary classification algorithm, and constructing a ranker from the binary classifier. Based on the framework, we show that a weighted 0/1 loss of the binary classifier upper-bounds the mislabeling cost of the ranker, both error-wise and regret-wise. Our framework allows not only to design good ordinal ranking algorithms based on well-tuned binary classification approaches, but also to derive new generalization bounds for ordinal ranking from known bounds for binary classification. In addition, our framework unifies many existing ordinal ranking algorithms, such as perceptron ranking and support vector ordinal regression. When compared empirically on benchmark data sets, some of our newly designed algorithms enjoy advantages in terms of both training speed and generalization performance over existing algorithms. In addition, the newly designed algorithms lead to better cost-sensitive ordinal ranking performance as well as improved listwise ranking performance.

1 Introduction

We work on a supervised learning problem called *ordinal ranking*, which is also referred to as ordinal regression (Chu and Keerthi, 2007) or ordinal classification (Frank and Hall, 2001). For instance, the rating that a customer gives on a movie might be one of `do-not-bother`, `only-if-you-must`, `good`, `very-good`, and `run-to-see`. Those ratings are called the *ranks*, which can be represented by ordinal class labels like $\{1, 2, 3, 4, 5\}$. The ordinal ranking problem is closely related to multi-class classification and metric regression. Somehow it is different from the former because of the ordinal information encoded in the ranks, and is different from the latter because of the lack of the metric distance between the ranks. Since rank is a natural representation of human preferences, the problem lends itself to many applications in social science and information retrieval (Liu, 2009).

Many ordinal ranking algorithms have been proposed from a machine learning perspective in recent years. For instance, Herbrich et al. (2000) designed an approach with support vector machines based on comparing training examples in a pairwise manner. Har-Peled et al. (2003) proposed a constraint classification approach that works with any binary classifiers based on the pairwise comparison framework. Nevertheless, such a pairwise comparison perspective may not be suitable for large-scale learning because the size of the associated optimization problem can be large. In particular, for an ordinal ranking problem with N examples, if at least two of the ranks are supported by $\Omega(N)$ examples (which is quite common in practice), the size of the pairwise learning problem is quadratic to N.

There are some other approaches that do not lead to such a quadratic expansion. For instance, Crammer and Singer (2005) generalized the online perceptron algorithm with multiple thresholds to do ordinal ranking. In their approach, a perceptron maps an input vector to a latent potential value, which is then thresholded to obtain a rank. Shashua and Levin (2003) proposed new support vector machine (SVM) formulations to handle multiple thresholds, and some other SVM formulations were studied by Rajaram et al. (2003); Chu and Keerthi (2007); Cardoso and da Costa (2007). All these algorithms share a common property: they are modified from well-known binary classification approaches. Still some other approaches fall into neither of the perspective above, such

as Gaussian process ordinal regression (Chu and Ghahramani, 2005).

Since binary classification is much better studied than ordinal ranking, a general framework to systematically reduce the latter to the former can introduce two immediate benefits. First, well-tuned binary classification approaches can be readily transformed into good ordinal ranking algorithms, which saves immense efforts in design and implementation. Second, new generalization bounds for ordinal ranking can be easily derived from known bounds for binary classification, which saves tremendous efforts in theoretical analysis.

In this paper, we propose such a reduction framework. The framework is based on extended binary examples, which are extracted from the original ordinal ranking examples. The binary classifier trained from the extended binary examples can then be used to construct a ranker. We prove that the mislabeling cost of the ranker is bounded by a weighted 0/1 loss of the binary classifier. Furthermore, we prove that the mislabeling regret of the ranker is bounded by the regret of the binary classifier as well. Hence, binary classifiers that generalize well could introduce rankers that generalize well. The advantages of the framework in algorithmic design and in theoretical analysis are both demonstrated in the paper. In addition, we show that our framework provides a unified view for many existing ordinal ranking algorithms. The experiments on some benchmark data sets validate the usefulness of the framework in practice, both for improving cost-sensitive ordinal ranking performance and for helping improve other ranking criteria.

The paper is organized as follows. We introduce the ordinal ranking problem in Section 2. Some related works are discussed in Section 3. We illustrate our reduction framework in Section 4. The algorithmic and theoretical usefulness of the framework is shown in Section 5. Finally, we present experimental results in Section 6, and conclude in Section 7.

A short version of the paper appeared in the 2006 Conference on Neural Information Processing Systems (Li and Lin, 2007b). The paper was then enriched by the more general cost-sensitive setting as well as the deeper theoretical results that were revealed in the 2009 Preference Learning Workshop (Lin and Li, 2009). For completeness, selected results from an earlier conference work (Lin and Li, 2006) are included in Subsection 5.2. The works above are also parts of the first author's Ph.D. thesis (Lin,

2008). In addition to the results that have been published in the conferences, we pointed out some important properties of ordinal ranking in Section 2, added a detailed literature discussion in Section 3, showed deeper theoretical results on the equivalence between ordinal ranking and binary classification in Section 4, clarified the differences of different SVM-based ordinal ranking algorithms in Section 5 and strengthened the experimental results to emphasize the usefulness of cost-sensitive ordinal ranking in Section 6.

2 Ordinal Ranking Setup

The ordinal ranking problem aims at predicting the rank y of some input vector $\mathbf{x}$, where $\mathbf{x}$ is in an input space $\mathcal{X} \subseteq \mathbb{R}^D$ and y is in a label space $\mathcal{Y} = \{1, 2, \cdots, K\}$. A function $r : \mathcal{X} \to \mathcal{Y}$ is called an *ordinal ranker*, or a *ranker* in short. We shall adopt the cost-sensitive setting (Abe et al., 2004; Lin, 2008), in which a cost vector $\mathbf{c} \in \mathbb{R}^K$ is generated with $(\mathbf{x}, y)$ from some fixed but unknown distribution $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ on $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}^K$. The k-th element $\mathbf{c}[k]$ of the cost vector represents the penalty when predicting the input vector $\mathbf{x}$ as rank k. We naturally assume that $\mathbf{c}[k] \geq 0$ and $\mathbf{c}[y] = 0$. Thus, $y = \text{argmin}_{1 \leq k \leq K} \mathbf{c}[k]$. An ordinal ranking problem comes with a given training set $\mathcal{S} = \{(\mathbf{x}_n, y_n, \mathbf{c}_n)\}_{n=1}^N$, whose elements are drawn i.i.d. from $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$. The goal of the problem is to find a ranker r such that its expected test cost

$$E(r) \equiv \mathop{\mathbb{E}}_{(\mathbf{x}, y, \mathbf{c}) \sim \mathcal{P}} \mathbf{c}[r(\mathbf{x})]$$

is small.

The setting above looks similar to that of a cost-sensitive multiclass classification problem (Abe et al., 2004), in the sense that the label space $\mathcal{Y}$ is a finite set. Therefore, ordinal ranking is also called ordinal classification (Frank and Hall, 2001; Cardoso and da Costa, 2007). Nevertheless, in addition to representing the nominal categories (as the usual classification labels), now those $y \in \mathcal{Y}$ also carry the ordinal information. That is, two different labels in $\mathcal{Y}$ can be compared by the usual "$<$" operation. Thus, those $y \in \mathcal{Y}$ are called the *ranks* to distinguish them from the usual classification labels.

Ordinal ranking is also similar to regression (for which $y \in \mathbb{R}$ instead of $\mathcal{Y}$), because the real values in $\mathbb{R}$ can be ordered by the usual "$<$" operation. Therefore, ordi-

nal ranking is also popularly called ordinal regression (Herbrich et al., 2000; Shashua and Levin, 2003; Chu and Ghahramani, 2005; Chu and Keerthi, 2007; Xia et al., 2007). Nevertheless, unlike the real-valued regression labels $y \in \mathbb{R}$, the discrete ranks $y \in \mathcal{Y}$ do not carry metric information. For instance, we cannot say that a five-star movie is 2.5 times better than a two-star one; we also cannot compute the exact distance (difference) between a five-star movie and a one-star movie. In other words, the rank serves as a qualitative indication rather than a quantitative outcome. Thus, any monotonic transform of the label space should not alter the ranking performance. Nevertheless, many regression algorithms depend on the assumed metric information and can be highly affected by monotonic transforms of the label space (which are equivalent to change-of-metric operations). Thus, those regression algorithms may not always perform well on ordinal ranking problems.

The ordinal information carried by the ranks introduces the following two properties, which are important for modeling ordinal ranking problems.

- **Closeness in the rank space** $\mathcal{Y}$: The ordinal information suggests that the mislabeling cost depend on the "closeness" of the prediction. For example, predicting a two-star movie as a three-star one is less costly than predicting it as a five-star one. Hence, the cost vector $\mathbf{c}$ should be *V-shaped* with respect to y (Li and Lin, 2007b), that is,

$$\begin{cases} \mathbf{c}[k-1] \geq \mathbf{c}[k]\,, & \text{for } 2 \leq k \leq y\,; \\ \mathbf{c}[k+1] \geq \mathbf{c}[k]\,, & \text{for } y \leq k \leq K-1. \end{cases} \tag{1}$$

  Briefly speaking, a V-shaped cost vector says that a ranker needs to pay more if its prediction on $\mathbf{x}$ is further away from y. We shall assume that every cost vector $\mathbf{c}$ generated from $\mathcal{P}(\mathbf{c} \,|\, \mathbf{x}, y)$ is V-shaped with respect to $y = \operatorname{argmin}_{1 \leq k \leq K} \mathbf{c}[k]$. In other words, one can decompose $\mathcal{P}(y, \mathbf{c} \,|\, \mathbf{x}) = \mathcal{P}(y \,|\, \mathbf{c})\, \mathcal{P}(\mathbf{c} \,|\, \mathbf{x})$ where $\mathbf{c} \sim \mathcal{P}(\mathbf{c} \,|\, \mathbf{x})$ is always V-shaped and $\mathcal{P}(y \,|\, \mathbf{c})$ satisfies $y = \operatorname{argmin}_{1 \leq k \leq K} \mathbf{c}[k]$.

  In some of our results, we need a stronger condition: the cost vectors should be *convex* (Li and Lin, 2007b), which is defined by the condition[1]

$$\mathbf{c}[k+1] - \mathbf{c}[k] \geq \mathbf{c}[k] - \mathbf{c}[k-1]\,, \text{ for } 2 \leq k \leq K-1\,. \tag{2}$$

[1] When connecting the points $(k, \mathbf{c}[k])$ from a convex cost vector $\mathbf{c}$ by line segments, it is not difficult to prove that the resulting curve is convex for $k \in [1, K]$.

When using convex cost vectors, a ranker needs to pay increasingly more if its prediction on $\mathbf{x}$ is further away from y. Provably, any convex cost vector $\mathbf{c}$ is V-shaped with respect to $y = \operatorname{argmin}_{1 \le k \le K} \mathbf{c}[k]$.

The V-shaped and convex cost vectors are general choices that can be used to represent the ordinal nature of $\mathcal{Y}$. One popular cost vector that has been frequently used for ordinal ranking is the absolute cost vector, which accompanies $(\mathbf{x}, y)$ as

$$\mathbf{c}^{(y)}[k] = |y - k|, \quad 1 \le y \le K.$$

Because the absolute cost vectors come with the $median$ function as its a population minimizer (Dembczyński et al., 2008), it appears to be a natural choice for ordinal ranking, similar to how the traditional 0/1 loss is the most natural choice for classification. Nevertheless, our work aims at studying more flexible possibilities (costs) beyond the natural choice, similar to the more flexible weighted loss beyond the 0/1 one in weighted classification (Zadrozny et al., 2003). As we shall show later in Section 6, the flexible costs can be used to embed the desired structural information in $\mathcal{Y}$ for better test performance.

- **Comparability in the input space** $\mathcal{X}$: Note that the classification cost vectors

$$\mathbf{c}^{(\ell)}[k] = [\![\ell \neq k]\!], \quad 1 \le \ell \le K,$$

which checks whether the predicted rank k is exactly the same as the desired rank ℓ, are also V-shaped.[2] If those cost vectors are used, an immediate question is: What distinguishes ordinal ranking and common multiclass classification?

Let r_* denote the optimal ranker with respect to $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$. Note that r_* introduces a total preorder in $\mathcal{X}$ (Herbrich et al., 2000). That is,

$$\mathbf{x} \lesssim \mathbf{x}' \iff r_*(\mathbf{x}) \le r_*(\mathbf{x}').$$

The total preorder allows us to naturally group and compare vectors in the input space $\mathcal{X}$. For instance, a two-star movie is "worse than" a three-star one, which is in turn "worse than" a four-star one; movies of less than three stars are "worse than" movies of at least three stars.

[2] $[\![\cdot]\!]$ is 1 if the inner condition is true, and 0 otherwise.

The *simplicity* of the grouping and the comparison distinguishes ordinal ranking from multiclass classification. For instance, when classifying movies, it is difficult to group $\{$action movies, romantic movies$\}$ and compare with $\{$comic movies, thriller movies$\}$, but "movies of less than three stars" can be naturally compared with "movies of at least three stars."

The comparability property connects ordinal ranking to monotonic classification (Sill, 1998; Kotłowski and Słowiński, 2009), which is also referred to as ordinal classification with the monotonicity constraints and is an important problem on its own. Monotonic classification models the ordinal ranking problem by assuming that an explicit order in the input space (such as the value-order of one particular feature) can be used to directly (and monotonically) infer about the order of the ranks in the output space ($y \leq y'$). In other words, monotonic classification allows putting thresholds on the explicit order to perform ordinal ranking. The comparability property shows that there is an order (total pre-order) introduced by the ranks. Nevertheless, the order is not always "explicit" in general ordinal ranking problems. Therefore, many of the existing ordinal ranking approaches, such as the thresholded model that will be discussed in Section 5, seek the implicit order through transforming the input vectors before respecting the monotonic nature between the implicit order and the order of the ranks.

In Table 1, we summarize four different learning problems in terms of their comparability and closeness properties.

Table 1: properties of different learning problems

closeness / comparability	weak (classification cost vectors)	strong (other V-shaped cost vectors)
yes	degenerate ordinal ranking	usual ordinal ranking
no	multiclass classification	special cases of cost-sensitive classification

As discussed, usual ordinal ranking problems come with strong closeness in $\mathcal{Y}$ (which is represented by V-shaped cost vectors) and simple comparability in $\mathcal{X}$. The

classification cost vectors can be viewed as degenerate V-shaped cost vectors, and hence introduce degenerate ordinal ranking problems.

Multiclass classification problems, on the other hand, do not allow examples of different classes to be naturally grouped and compared. If we want to use cost vectors other than the classification ones, we move to special cases of cost-sensitive classification. For instance, when trying to recognize digits $\{0, 1, \cdots, 9\}$ for written checks, a possible cost is the absolute one (to represent monetary differences) rather than simply right or wrong (the classification cost). The absolute cost is V-shaped and convex. Nevertheless, the digits intuitively cannot be grouped and compared, and hence the recognition problem belongs to cost-sensitive multiclass classification rather than ordinal ranking (Lin, 2008).

From the discussions above, a good ordinal ranking algorithm should appropriately use the comparability property. In Section 4, we will show how the property serves as a key to derive our proposed reduction framework.

3 Related Literature

The analysis of ordinal data has been studied in statistics by defining a suitable link function that models the underlying probability for generating the ordinal labels (Anderson, 1984). For instance, one popular model is the the cumulative link model (Agresti, 2002) that will be discussed in Section 5. Similar models can be traced back to the work of McCullagh (1980). The many earlier works in statistics, which usually focus on the effectiveness and efficiency of the modeling, influence the ordinal ranking studies in machine learning (Herbrich et al., 2000), including our work. Another related area that study the analysis of ordinal data is operations research, especially in the subarea of multi-criteria decision analysis (Greco et al., 2000; Figueira et al., 2005), which contains many works that focus on reasonable decision making with ordinal preference scales. Our work tackles ordinal ranking problems from the machine learning perspective—improving the test performance—and is hence different from the works that take the perspective of statistics or operations research.

In machine learning (and information retrieval), there are three major families of ranking algorithms: pointwise, pairwise and listwise (Liu, 2009). The ordinal ranking

setup presented in Section 2 belongs to pointwise ranking. Next, we discuss some representative algorithms in each family and relate them to the ordinal ranking setup. Then, we compare the proposed reduction framework with other reduction-based approaches for ranking.

3.1 Families of Ranking Algorithms

Pointwise ranking. Pointwise ranking aims at predicting the relevance of some input vector $\mathbf{x}$ using either real-valued scores or ordinal-valued ranks. It does not directly use the comparison nature of ranking.

The ordinal ranking algorithms studied in this paper focus on computing ordinal-valued ranks for pointwise ranking. For obtaining real-valued scores, a fundamental tool is traditional least-squared regression (Hastie et al., 2001). As discussed in Section 2, however, when the given examples come with ordinal labels, the ordinal ranking algorithms studied in this paper can be more useful than traditional regression by taking the metric-less nature of labels into account.

Pairwise ranking. Pairwise ranking aims at predicting the relative order between two input vectors $\mathbf{x}$ and $\mathbf{x}'$ and thus captures the local comparison nature of ranking. It is arguably one of the most widely used technique in the ranking family and is usually cast as a binary classification problem of predicting whether $\mathbf{x}$ is preferred over $\mathbf{x}'$. During training, such a problem translates to comparing all pairs of $(\mathbf{x}_n, \mathbf{x}_m)$ based on their corresponding labels. One representative pairwise ranking algorithm is RankSVM (Herbrich et al., 2000), which trains an underlying support vector machine using those pairs. RankSVM was initially proposed for data sets that come with ordinal labels, but is also commonly applied to data sets that come with real-valued labels.

Note that even when all the labels takes ordinal values, as long as two of the classes contain $\Omega(N)$ examples, there are $\Omega(N^2)$ pairs. Such a quadratic number of pairs makes it difficult to scale up general pairwise ranking algorithms, except in special cases like the linear support vector machine (Joachims, 2006) or RankBoost (Herbrich et al., 2000; Lin and Li, 2006). Thus, when the training set is large and contains ordinal labels, the ordinal ranking algorithms studied in this paper may serve as a useful alternative over

pairwise ranking ones.

Listwise ranking. Listwise ranking aims at ordering a whole finite set of input vectors $\mathcal{S}' = \{\mathbf{x}'_m\}_{m=1}^M$. In particular, the (listwise) ranker tries to minimize the inconsistency between the predicted permutation and the ground truth permutation of $\mathcal{S}'$ (Liu, 2009). Listwise ranking captures the global comparison nature of ranking. One representative listwise ranking algorithm is ListNet (Cao et al., 2007), which is based on an underlying neural network model along with an estimated distribution of all possible permutations (rankers). Nevertheless, there are $M!$ permutations for a given $\mathcal{S}'$. Thus, listwise ranking can be computationally even more expensive than pairwise ranking.

Many listwise ranking algorithms try to alleviate the computational burden by keeping some internal pointwise rankers. For instance, ListNet uses the underlying neural network to score each instance (Cao et al., 2007) for the purpose of permutation. The use of internal pointwise rankers for listwise ranking further justify the importance to better understand pointwise ranking, including the ordinal ranking algorithms studied in this paper.

3.2 Reduction Approaches for Ranking

Because ranking is a relatively new and diverse problem in machine learning, many existing ranking approaches try to reduce the ranking problem to other learning problems. Next, we discuss some existing reduction-based approaches that are related to the framework proposed in this paper.

From pairwise ranking to binary classification. Balcan et al. (2007) propose a robust reduction from bipartite (i.e. ordinal with two outcomes) pairwise ranking to binary classification. The training part of the reduction works like usual pairwise ranking: learning a binary classifier on whether $\mathbf{x}$ is preferred over $\mathbf{x}'$. The prediction part of the reduction asks the underlying binary classifier to vote for each example in the test set in order to rank those examples. The reduction is simple but yields solid theoretical guarantees. In particular, for ranking M test examples, the reduction uses $\Omega(M^2)$ calls to the binary classifier and transforms a binary classification regret of r to a bipartite ranking regret (measured by the so-called AUC criterion) of at most $2r$.

Ailon and Mohri (2008) improve the reduction of Balcan et al. (2007) and propose a more efficient reduction from general pairwise ranking to binary classification. The prediction part of the reduction operates by taking the underlying binary classifier as the comparison function of the popular QuickSort algorithm. In the special bipartite ranking case, for ranking M examples, the reduction uses $O(M \log M)$ calls to the binary classifier in average and transforms a binary classification regret of r to a bipartite ranking regret of at most $2r$.

From listwise ranking to regression (pointwise ranking). The Subset Ranking (Cossock and Zhang, 2008) algorithm can be viewed as a reduction from listwise ranking to regression. In particular, Cossock and Zhang (2008) prove that regression with various cost functions can be used to approximate a Bayes optimal listwise ranker. In other words, low-regret regressors can be cast as low-regret listwise rankers.

From listwise ranking to ordinal (pointwise) ranking. McRank (Li et al., 2008) is a reduction from listwise ranking to ordinal ranking with the classification cost. The main theoretical justification of the reduction shows that a scaled classification cost of an ordinal ranker can upper bound the regret of the associated listwise ranker. That is, low-error ordinal rankers can be cast as low-regret listwise rankers. Li et al. (2008) empirically verified that McRank can perform better than the Subset Ranking algorithm (Cossock and Zhang, 2008).

From ordinal ranking to binary classification. The proposed framework in this paper and the associated shorter version (Li and Lin, 2007b) is a reduction from ordinal ranking to binary classification. We will show that the reduction is both error and regret preserving. That is, low-error binary classifiers can be cast as low-error ordinal rankers; low-regret binary classifiers can be cast as low-regret ordinal rankers.

The data replication method, which was independently proposed by Cardoso and da Costa (2007), is a similar but more restricted case of the reduction framework. The data replication method essentially considers the absolute cost. In addition, the focus of the data replication method (Cardoso and da Costa, 2007) is on explaining the training procedure of the reduction. The proposed framework in this paper is more general than the data replication method in terms of the cost considered as well as the deeper

Table 2: comparison of general reductions from ranking to binary classification

reduction	size of transformed set during training	# calls to binary classifiers during prediction	evaluation criterion
the proposed framework	$O(KN)$	$O(KM)$	ranking cost
(Balcan et al., 2007)	$O(N^2)$	$O(M^2)$	AUC
(Ailon and Mohri, 2008)	$O(N^2)$	$O(M \log M)$	AUC

theoretical analysis on both the training and the test performance of the reduction.

The proposed reduction framework for pointwise ranking and existing reductions in pairwise ranking (Balcan et al., 2007; Ailon and Mohri, 2008) take very different views on the ranking problem and considers different evaluation criteria. As a consequence, when learning N examples and ranking (predicting on) M instances with K ordinal scales, the proposed framework results in a transformed training set of size $O(KN)$ and a prediction procedure with time complexity $O(KM)$. Both the size of the training set and the time complexity of the prediction procedure is more efficient than the state-of-the-art reduction from pairwise ranking to binary classification (Ailon and Mohri, 2008), as shown in Table 2.

Note that the work of Li et al. (2008) revealed an opportunity to use the discrete nature of ordinal-valued labels to improve the listwise ranking performance over Subset Ranking when using a heuristic ordinal ranking algorithm. The proposed framework is a more rigorous study on ordinal ranking that can be coupled with McRank to yield a reduction from listwise ranking to binary classification, which allows state-of-art binary classification algorithms to be efficiently used for listwise ranking. We will demonstrate the use of this opportunity in Subsection 6.4.

4 Reduction Framework

We will first introduce the details of our proposed reduction framework. Then, we will demonstrate its theoretical guarantees. Consider, for instance, that we want to know how good a movie $\mathbf{x}$ is. Using the comparability property of ordinal ranking, we can then ask the associated question "is the rank of $\mathbf{x}$ greater than k?"

For a given k, such a question is exactly a binary classification problem, and the rank of $\mathbf{x}$ can be determined by asking multiple questions for $k = 1, 2$, until $(K-1)$. The questions are the core of the Dominance-based Rough Set Approach in operations research for reasoning from ordinal data (Słowiński et al., 2007). From the machine learning perspective, Frank and Hall (2001) proposed to solve each binary classification problem independently and combine the binary outputs to a rank. Although their approach is simple, the generalization performance using the combination step cannot be easily analyzed.

The proposed reduction framework works differently. First, a simpler step is used to convert the binary outputs to a rank, and generalization analysis can immediately follow. Moreover, all the binary classification problems are solved jointly to obtain a single binary classifier.

Assume that $g(\mathbf{x}, k)$ is the single binary classifier that provides answers to all the associated questions above. *Consistent* answers would be $g(\mathbf{x}, k) = +1$ ("yes") for $k = 1$ until $(\ell - 1)$ for some ℓ, and -1 ("no") afterward. Then, a reasonable ranker based on the binary answers is $r_g(\mathbf{x}) = \ell = 1 + \min\{k\colon g(\mathbf{x}, k) = +1\}$. Equivalently,

$$r_g(\mathbf{x}) \equiv 1 + \sum_{k=1}^{K-1} [\![g(\mathbf{x}, k) > 0]\!] \,. \tag{3}$$

The binary classifier g that only produces consistent answers would be called *rank-monotonic*.[3]

For any ordinal example $(\mathbf{x}, y, \mathbf{c})$, we can define the extended binary examples $\left(\mathbf{X}^{(k)}, Y^{(k)}\right)$ with weights $W^{(k)}$ as

$$\mathbf{X}^{(k)} = (\mathbf{x}, k), \quad Y^{(k)} = 2\,[\![k < y]\!] - 1, \quad W^{(k)} = (K-1) \cdot \left| \mathbf{c}[k] - \mathbf{c}[k+1] \right| \tag{4}$$

The extended input vector $\mathbf{X}^{(k)}$ represents the associated question "is the rank of $\mathbf{x}$ greater than k?"; the binary label $Y^{(k)}$ represents of the desired answer to the question; the weight $W^{(k)}$ represents the importance of the question and will be used in the coming theoretical analysis. Here $\mathbf{X}^{(k)}$ stands for an abstract pair and we will discuss its practical encoding in Section 5. If $g\left(\mathbf{X}^{(k)}\right) \equiv g(\mathbf{x}, k)$ makes no errors on all the associated questions, $r_g(\mathbf{x})$ equals y by (3). That is, $\mathbf{c}[r_g(\mathbf{x})] = 0$. In the following theorem, we further connects $\mathbf{c}[r_g(\mathbf{x})]$ to the amount of error that g makes.

[3] Although (3) can be flexibly applied even when g is not rank-monotonic, a rank-monotonic g is usually desired in order to introduce a good ranker r_g.

Theorem 1 (Per-example cost bound)**.** *For any ordinal example* $(\mathbf{x}, y, \mathbf{c})$*, where* $\mathbf{c}$ *is V-shaped and* $\mathbf{c}[y] = 0$*, consider its associated extended binary examples* $\big(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\big)$ *in* (4)*. Assume that the ranker* r_g *is constructed from a binary classifier* g *using* (3)*. If* $g\big(\mathbf{X}^{(k)}\big)$ *is rank-monotonic or if* $\mathbf{c}$ *is convex, then*

$$\mathbf{c}[r_g(\mathbf{x})] \;\leq\; \frac{1}{K-1} \sum_{k=1}^{K-1} W^{(k)} \cdot \big[\!\big[Y^{(k)} \neq g\big(\mathbf{X}^{(k)}\big) \big]\!\big] \,. \tag{5}$$

Proof. Because g is rank-monotonic, $g\big(\mathbf{X}^{(k)}\big) = +1$ for $k < r_g(\mathbf{x})$ and $g\big(\mathbf{X}^{(k)}\big) = -1$ for $k \geq r_g(\mathbf{x})$. Thus, the cost that the ranker r_g needs to pay is

$$\begin{aligned} \mathbf{c}[r_g(\mathbf{x})] \;&=\; \sum_{k=r_g(\mathbf{x})}^{K-1} (\mathbf{c}[k] - \mathbf{c}[k+1]) + \mathbf{c}[K] \\ &=\; \sum_{k=1}^{K-1} (\mathbf{c}[k] - \mathbf{c}[k+1]) \, \big[\!\big[g\big(\mathbf{X}^{(k)}\big) < 0 \big]\!\big] + \mathbf{c}[K] \,. \end{aligned} \tag{6}$$

Because the cost vector $\mathbf{c}$ is V-shaped, $Y^{(k)}$ equals the sign of $(\mathbf{c}[k] - \mathbf{c}[k+1])$ if the latter is not zero. Continuing from (6) with $\mathbf{c}[y] = 0$,

$$\begin{aligned} & (K-1) \cdot \mathbf{c}[r_g(\mathbf{x})] \\ =\; & \sum_{k=1}^{y-1} W^{(k)} \cdot Y^{(k)} \cdot \big[\!\big[g\big(\mathbf{X}^{(k)}\big) < 0 \big]\!\big] + (K-1) \cdot \mathbf{c}[K] \\ & \sum_{k=y}^{K-1} W^{(k)} \cdot Y^{(k)} \cdot \big(1 - \big[\!\big[g\big(\mathbf{X}^{(k)}\big) > 0 \big]\!\big]\big) \\ =\; & \sum_{k=1}^{y-1} W^{(k)} \cdot \big[\!\big[Y^{(k)} \neq g\big(\mathbf{X}^{(k)}\big) \big]\!\big] + (K-1) \cdot \mathbf{c}[y] + \sum_{k=y}^{K-1} W^{(k)} \cdot \big[\!\big[Y^{(k)} \neq g\big(\mathbf{X}^{(k)}\big) \big]\!\big] \\ =\; & \sum_{k=1}^{K-1} W^{(k)} \cdot \big[\!\big[Y^{(k)} \neq g\big(\mathbf{X}^{(k)}\big) \big]\!\big] \,. \end{aligned} \tag{7}$$

When g is not rank-monotonic but the cost vector $\mathbf{c}$ is convex, equation (7) becomes an inequality that could be alternatively proved by replacing (6) with

$$\sum_{k=r_g(\mathbf{x})}^{K-1} (\mathbf{c}[k] - \mathbf{c}[k+1]) \leq \sum_{k=1}^{K-1} (\mathbf{c}[k] - \mathbf{c}[k+1]) \, \big[\!\big[g\big(\mathbf{X}^{(k)}\big) < 0 \big]\!\big] \,.$$

The inequality above holds because $(\mathbf{c}[k] - \mathbf{c}[k+1])$ is decreasing due to the convexity, and there are exactly $(r_g(\mathbf{x}) - 1)$ zeros and $(K - r_g(\mathbf{x}))$ ones in the values of $\big[\!\big[g\big(\mathbf{X}^{(k)}\big) < 0 \big]\!\big]$ according to (3). □

We call (5) the per-example cost bound, which says that if g makes only a small amount of error on the extended binary examples $\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$, then r_g is guaranteed to only pay a small amount of cost on the original example $(\mathbf{x}, y, \mathbf{c})$. The bound allows us to derive the following reduction method, which is composed of three stages: preprocessing, training, and prediction.

Algorithm 1 (Reduction to extended binary classification)**.**

1. *Preprocessing: For each original training example* $(\mathbf{x}_n, y_n, \mathbf{c}_n) \in \mathcal{S}$ *and for each* $k = 1, 2, \ldots, K-1$*, generate an extended training example* $\left(\mathbf{X}_n^{(k)}, Y_n^{(k)}, W_n^{(k)}\right)$ *and include it in* $\mathcal{S}_E$*, where*

$$\mathbf{X}_n^{(k)} = (\mathbf{x}_n, k),\ Y_n^{(k)} = 2\left[\!\left[k < y_n\right]\!\right] - 1,\ W_n^{(k)} = (K-1) \cdot \left|\mathbf{c}_n[k] - \mathbf{c}_n[k+1]\right|.$$

2. *Training: Use a binary classification algorithm on* $\mathcal{S}_E$ *and get a binary classifier* g *on a concrete encoding (to be discussed in Section 5) of* $\mathcal{X} \times \{1, 2, \cdots, K-1\}$*. Let* $g(\mathbf{x}, k) \equiv g\left(\mathbf{X}^{(k)}\right)$.

3. *Prediction: For any* $\mathbf{x} \in \mathcal{X}$*, estimate its rank with* (3)*.*

4.1 Cost Bound of the Reduction Framework

Consider the following probability distribution $\mathcal{P}_b\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$ that generates the extended binary examples.

1. Draw a tuple $(\mathbf{x}, y, \mathbf{c})$ independently from $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ and draw k uniformly from the set $\{1, 2, \ldots, K-1\}$.

2. Generate $\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$ by (4).

The extended training set S_E contains examples that are equivalent (in terms of expectation) to examples drawn independently from $\mathcal{P}_b\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$. For any given binary classifier g, define its out-of-sample error with respect to $\mathcal{P}_b$ as

$$E_b(g) \equiv \mathop{\mathbb{E}}_{(\mathbf{X},Y,W)\sim\mathcal{P}_b} W \cdot \left[\!\left[Y \neq g(\mathbf{X})\right]\!\right].$$

Using the definitions above, we can prove the first theoretical guarantee of the reduction framework.

Theorem 2 (Cost bound of the reduction framework)**.** *Consider a ranker r_g that is constructed from a binary classifier g using* (3)*. Assume that* $\mathbf{c}$ *is V-shaped and* $\mathbf{c}[y] = 0$ *for every tuple* $(\mathbf{x}, y, \mathbf{c})$ *generated from* $\mathcal{P}(\mathbf{c} \mid \mathbf{x}, y)$*. If* $g(\mathbf{x}, k)$ *is rank-monotonic or if every cost vector* $\mathbf{c}$ *is convex, then* $E(r_g) \leq E_b(g)$*.*

Proof. From (5),

$$\mathbf{c}[r_g(\mathbf{x})] \;\leq\; \frac{1}{K-1} \sum_{k=1}^{K-1} W^{(k)} \cdot [\![Y^{(k)} \neq g(\mathbf{X}^{(k)})]\!] \,.$$

Take the expectation over $\mathcal{P}$ on both sides and use $\sim_u$ to mean the uniform sampling,

$$\begin{aligned} E(r_g) &\leq \mathop{\mathbb{E}}_{(\mathbf{x},y,\mathbf{c})\sim\mathcal{P}} \frac{1}{K-1} \sum_{k=1}^{K-1} W^{(k)} \cdot [\![Y^{(k)} \neq g(\mathbf{X}^{(k)})]\!] \\ &= \mathop{\mathbb{E}}_{(\mathbf{x},y,\mathbf{c})\sim\mathcal{P}} \mathop{\mathbb{E}}_{k\sim_u\{1,\cdots,K-1\}} W^{(k)} \cdot [\![Y^{(k)} \neq g(\mathbf{X}^{(k)})]\!] \\ &= \mathop{\mathbb{E}}_{(\mathbf{X},Y,W)\sim\mathcal{P}_b} W \cdot [\![Y \neq g(\mathbf{X})]\!] \\ &= E_b(g). \end{aligned}$$

□

4.2 Regret Bound of the Reduction Framework

Theorem 2 indicates that if there exists a decent binary classifier g, we can obtain a decent ranker r_g. Nevertheless, it does not guarantee how good r_g is in comparison with other rankers. In particular, if we consider the optimal binary classifier g_* under $\mathcal{P}_b(\mathbf{X}, Y, W)$, and the optimal ranker r_* under $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$, does a small regret $E_b(g) - E_b(g_*)$ in binary classification translate to a small regret $E(r_g) - E(r_*)$ in ordinal ranking? Furthermore, is $E(r_{g_*})$ close to $E(r_*)$? Next, we introduce the *reverse reduction technique*, which helps to answer the questions above.

The reverse reduction technique works on the binary classification problems generated by the reduction method. It goes through the preprocessing and the prediction stages of the reduction method in the opposite direction. In the preprocessing stage, instead of starting with ordinal examples $(\mathbf{x}_n, y_n, \mathbf{c}_n)$, reverse reduction deals with weighted binary examples $\left(\mathbf{X}_n^{(k)}, Y_n^{(k)}, W_n^{(k)}\right)$. It first combines each set of binary

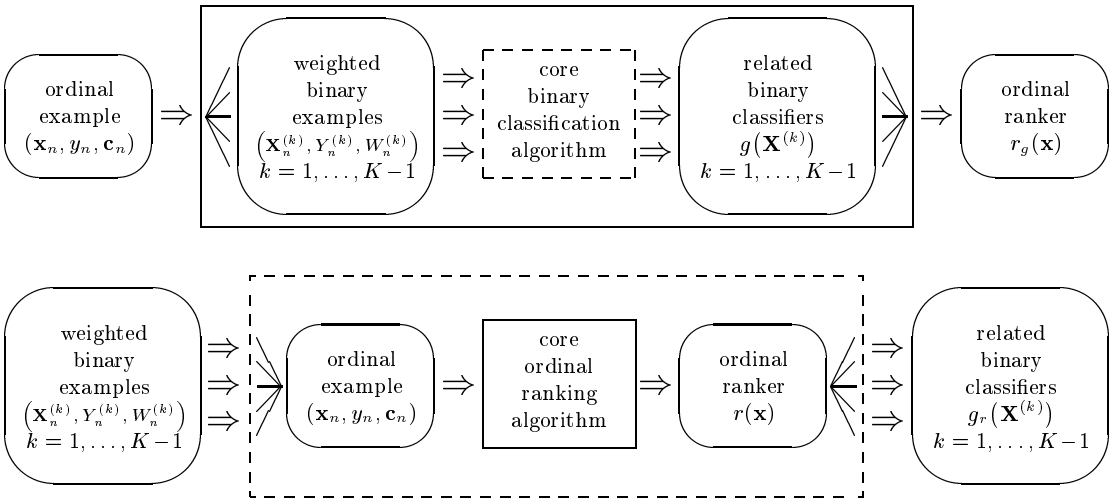

Figure 1: reduction (top) and reverse reduction (bottom)

examples sharing the same $\mathbf{x}_n$ to an ordinal example by

$$\begin{cases} y_n = & 1 + \sum_{k=1}^{K-1} \left[\!\left[Y_n^{(k)} > 0 \right]\!\right] \ ; \\ \mathbf{c}_n[k] = & \sum_{\ell=1}^{K-1} \frac{W_n^{(\ell)}}{K-1} \cdot \left[\!\left[y_n \le \ell < k \text{ or } k < \ell \le y_n \right]\!\right]. \end{cases} \tag{8}$$

It is easy to verify that (8) is the exact inverse transform of (4) on the training examples under the assumption that $\mathbf{c}[y] = 0$. These ordinal examples are then given to an ordinal ranking algorithm to obtain a ranker r. In the prediction stage, reverse reduction works by decomposing the prediction $r(\mathbf{x})$ to $K-1$ binary predictions, each as if coming from a binary classifier

$$g_r\left(\mathbf{X}^{(k)}\right) = 2\left[\!\left[r(\mathbf{x}) > k \right]\!\right] - 1. \tag{9}$$

Then, a lemma on the out-of-sample cost of g_r immediately follows (Lin and Li, 2009).

Lemma 1. *With the definitions of $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ and $\mathcal{P}_b\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$ in Theorem 2, for every ordinal ranker r, $E(r) = E_b(g_r)$.*

Proof. Because g_r is rank-monotonic by construction, the same proof for the first part of Theorem 2 leads to the desired result. □

The stages of reduction and reverse reduction are illustrated in Figure 1. Next, we show how the reverse reduction technique allows us to draw a strong theoretical

connection between ordinal ranking and binary classification. By the definition of r_* and g_*, for any ranker r and any binary classifier g,

$$E(r) \geq E(r_*), \quad E_b(g) \geq E_b(g_*) \ . \tag{10}$$

Then, the reverse reduction technique yields a simple proof of the regret bound.

Theorem 3 (Regret bound of the reduction framework)**.** *If $g(\mathbf{x}, k)$ is rank-monotonic, or if every cost vector* $\mathbf{c}$ *is convex, then*

$$E(r_g) - E(r_*) \leq E_b(g) - E_b(g_*). \tag{11}$$

Proof.

$$\begin{aligned} E(r_g) - E(r_*) \quad &\leq \quad E_b(g) - E(r_*) \qquad (\text{from Theorem 2}) \\ &= \quad E_b(g) - E_b(g_{r_*}) \qquad (\text{from Lemma 1}) \\ &\leq \quad E_b(g) - E_b(g_*) \qquad \big(\text{from Equation 10}\big) \quad . \end{aligned}$$

□

The cost bound (Theorem 2) and the regret bound (Theorem 3) provide different guarantees for the reduction method. The former describes how the ordinal ranking cost is upper bounded by the binary classification error in an absolute sense, and the latter describes the upper bound in a relative sense.

4.3 Equivalence between Ordinal Ranking and Binary Classification

The results above suggest that ordinal ranking can be reduced to binary classification without any loss of optimality. That is, ordinal ranking is "no harder than" binary classification. Intuitively, binary classification is also "no harder than" ordinal ranking, because the former is a special case of the latter with $K = 2$. Next, we formalize the notion of hardness with the *probably approximately correct* (PAC) setup in computational learning theory (Kearns and Vazirani, 1994) and prove that ordinal ranking and binary classification are indeed equivalent in hardness. We use the following definition of PAC in our coming theorems (Valiant, 1984; Kearns and Vazirani, 1994).

Definition 1. *In cost-sensitive classification, a learning model $\mathcal{G}$ is efficiently PAC-learnable (using the same representation class) if there exists a (possibly randomized) learning algorithm $\mathcal{A}$ satisfying the following property: for every distribution $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ being considered, where*

$$\mathbf{c}[g_*(\mathbf{x})] = \mathbf{c}[y] = c_{\min} = 0,$$

with some $g_* \in \mathcal{G}$; for all $0 < \epsilon$ and $0 < \delta < \frac{1}{2}$, if $\mathcal{A}$ is given access to an oracle generating examples $(\mathbf{x}, y, \mathbf{c})$ from $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$, as well as inputs ϵ and δ, then $\mathcal{A}$ outputs $g \in \mathcal{G}$ such that $E(g) \leq \epsilon$ with probability at least $1 - \delta$ as well as with time polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

Briefly speaking, the definition assumes that the target function g_* is within the learning model $\mathcal{G}$ and is of cost 0 (the minimum cost). In other words, it is the noiseless setup of learning. We shall only focus on this case while pointing out that similar results can also be proved for the noisy setup (Lin, 2008).

Theorem 4 (Equivalence theorem of the reduction framework)**.** *Consider a learning model $\mathcal{R}$ for ordinal ranking, its associated learning model $\mathcal{G} = \{g_r \colon r \in \mathcal{R}\}$ for binary classification, and distributions $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ such that all cost vectors $\mathbf{c}$ are V-shaped. Then, $\mathcal{R}$ is efficiently PAC-learnable if and only if $\mathcal{G}$ is efficiently PAC-learnable.*

Proof. If $\mathcal{G}$ is efficiently PAC-learnable using algorithm $\mathcal{A}_{\mathcal{G}}$, we can convert $\mathcal{A}_{\mathcal{G}}$ to an efficient algorithm $\mathcal{A}_{\mathcal{R}}$ for ordinal ranking as follows.

1. Transform the oracle that generates $(\mathbf{x}, y, \mathbf{c})$ from $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ to an oracle that generates $\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$ by picking k uniformly and applying (4).

2. Run $\mathcal{A}_{\mathcal{G}}$ with the transformed oracle until it outputs some $g\left(\mathbf{X}^{(k)}\right)$.

3. Return r_g.

It is not hard to see that $\mathcal{A}_{\mathcal{R}}$ is as efficient as $\mathcal{A}_{\mathcal{G}}$, and the cost guarantee comes from Theorem 2 using the fact that g_r are all rank-monotonic.

Now we consider the other direction. If $\mathcal{R}$ is efficiently PAC-learnable using algorithm $\mathcal{A}_{\mathcal{R}}$, we can convert $\mathcal{A}_{\mathcal{R}}$ to an efficient algorithm $\mathcal{A}_{\mathcal{G}}$ for binary classification.

1. Transform the oracle that generates $\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$ from $\mathcal{P}_b\left(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)}\right)$ to an oracle that generates $(\mathbf{x}, y, \mathbf{c})$ by

$$\begin{aligned}
\mathbf{x} &= \left(\mathbf{X}^{(k)}[1], \mathbf{X}^{(k)}[2], \ldots, \mathbf{X}^{(k)}[D]\right); \\
\mathbf{c} &= \begin{cases} \frac{W^{(k)}}{K-1} \cdot \left(\underbrace{0, \ldots, 0}_{k}, 1, \ldots, 1\right) & \text{for } Y^{(k)} = -1, \\ \frac{W^{(k)}}{K-1} \cdot \left(\underbrace{1, \ldots, 1}_{k}, 0, \ldots, 0\right) & \text{for } Y^{(k)} = +1; \end{cases} \\
y &= \operatorname*{argmin}_{1 \le \ell \le K} \mathbf{c}[\ell], \text{ with ties arbitrarily broken.}
\end{aligned}$$

   That is, $\mathbf{x}$ copies the 1-st to the D-th elements of $\mathbf{X}^{(k)}$. Let $\tilde{\mathcal{P}}(\mathbf{x}, y, \mathbf{c})$ be the underlying distribution of the constructed oracle.

2. Run $\mathcal{A}_{\mathcal{R}}$ with the transformed oracle until it outputs some $r(\mathbf{x})$.

3. Return g_r.

Note that $\mathcal{A}_{\mathcal{G}}$ is as efficient as $\mathcal{A}_{\mathcal{R}}$. In addition, we see that plugging $\tilde{\mathcal{P}}$ into (4) introduces $\mathcal{P}_b$. Thus, if we take $\tilde{E}(r)$ as the expected test cost with respect to $\tilde{\mathcal{P}}$, by Lemma 1,

$$E_b(g_r) = \tilde{E}(r) \text{ for all } r \in \mathcal{R}.$$

Therefore, $E_b(g_r) < \epsilon$ after running $\mathcal{A}_{\mathcal{G}}$. □

Theorem 4 demonstrates that ordinal ranking is theoretically as easy (hard) as the associated binary classification problem. Recall that we compare four different kinds of learning problems in Table 1 of Section 2. At first sight, Theorem 4 appears to suggest that all four problems can be conquered with the reduction framework, because the only required assumption of the theorem is that the cost vectors are V-shaped. Nevertheless, note that the necessary and sufficient condition in the theorem is "*the associated learning model $\mathcal{G}$ is efficiently PAC-learnable*." Then, the different comparability properties of the different problems make a difference. In particular, for multi-class classification problems, the associated binary question "is the *class* of $\mathbf{x}$ greater than k?" can be complicated and is thus difficult to learn, In other words, the associated $\mathcal{G}$ may not be efficiently PAC-learnable. Then, more complicated binary questions (Abe et al., 2004;

Beygelzimer et al., 2005, 2007; Lin, 2008) are needed to reduce from the general (cost-sensitive) multiclass problem to binary classification ones.

On the other hand, for the special case of cost-sensitive ordinal ranking, in which $\mathcal{G}$ is efficiently PAC-learnable, the reduction framework establishes a tight connection between the learnability of $\mathcal{G}$ and $\mathcal{R}$—the ranking model of interest. The tight connection motivates us to design ordinal ranking algorithms from popular binary classification algorithms, as shown in the next section.

5 Applications of Reduction Framework

So far the reduction works only by assuming that $\mathbf{X}^{(k)} = (\mathbf{x}, k)$ is an abstract pair understandable by the binary classification algorithm. With proper choices of the cost vectors, the encoding scheme of $(\mathbf{x}, k)$, and the binary classification algorithm, many existing ordinal ranking algorithms can be unified in our framework, and their theoretical justifications can immediately follow.

In this section, we will briefly discuss some of those algorithms and their theoretical justifications. It happens that a simple encoding scheme for $(\mathbf{x}, k)$ via a coding matrix $\mathbf{M}$ of $(K-1)$ rows works for all the algorithms. To form $\mathbf{X}^{(k)}$, the vector $\mathbf{m}_k$, which denotes the k-th row of $\mathbf{M}$, is appended after $\mathbf{x}$. We will mostly work with $\mathbf{M} = \gamma \cdot \mathbf{I}_{K-1}$, where γ is a positive scalar and $\mathbf{I}_{K-1}$ is the $(K-1) \times (K-1)$ identity matrix.

5.1 Perceptron for Ordinal Ranking

The perceptron ranking (PRank) algorithm proposed by Crammer and Singer (2005) is an online ordinal ranking algorithm that employs the thresholded linear model

$$r(\mathbf{x}) = \min \{k : \langle \mathbf{v}, \mathbf{x} \rangle \leq \theta_k\},$$

where the thresholds $\theta_1, \theta_2, \cdots, \theta_{K-1}, \theta_K$ are ordered such that $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_{K-1} \leq \theta_K = \infty$. Whenever a training example is not predicted correctly, the current $\mathbf{v}$ and $\boldsymbol{\theta}$ are updated in a way similar to the perceptron learning rule (Rosenblatt, 1962). The algorithm was proved to keep the thresholds ordered along with a mistake bound (Crammer and Singer, 2005).

Let $\mathbf{X}^{(k)} = (\mathbf{x}, \mathbf{m}_k)$ with the simple encoding scheme $\mathbf{M} = \mathbf{I}_{K-1}$. Then,

$$r(\mathbf{x}) = \min\{k\colon \langle \mathbf{v}, \mathbf{x}\rangle \leq \theta_k\} = 1 + \sum_{k=1}^{K-1} [\![\langle (\mathbf{v}, -\boldsymbol{\theta}), \mathbf{X}^{(k)} \rangle > 0]\!] .$$

Consider an ordinal ranking problem such that $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ only generates examples $\left(\mathbf{x}, y, \mathbf{c}^{(y)}\right)$ where $\mathbf{c}^{(y)}$ is the absolute cost vector with respect to y. We see that $W^{(k)} = K-1$ (a constant) for all the extended binary examples. Then, we can simply interpret PRank as a specific instance of the reduction framework with a modified perceptron learning rule as the underlying binary classification algorithm. That is, PRank uses the perceptron learning rule to find a weight vector $(\mathbf{v}, -\boldsymbol{\theta})$ for classifying the extended binary examples $(\mathbf{x}, \mathbf{m}_k)$.[4] The mistake bound is a direct application of the well-known perceptron mistake bound (see, for example, Freund and Schapire, 1999). Our framework not only simplifies the derivation of the mistake bound, but also allows the use of other underlying perceptron algorithms, such as batch-mode algorithms (Li and Lin, 2007a) rather than online ones.

5.2 Boosting for Ordinal Ranking

In our earlier work (Lin and Li, 2006), we proposed the thresholded ensemble model

$$r(\mathbf{x}) = \min\{k\colon H_T(\mathbf{x}) \leq \theta_k\}, \text{ where } H_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}), \tag{12}$$

for ordinal ranking. Each confidence function $h_t\colon \mathcal{X} \to [-1, +1]$ reflects a possibly imperfect ordering preference. Note that a special instance of the confidence function is a binary classifier $\mathcal{X} \to \{-1, +1\}$. The ensemble linearly combines the ordering preferences with α. We allow α_t to be any real value, which means that it is possible to reverse the ordering preference of h_t in the ensemble when necessary.

Ensemble models in general have been successfully used for classification and regression (Meir and Rätsch, 2003). They not only introduce more stable predictions through the linear combination, but also provide sufficient power for approximating complicated target functions. The thresholded ensemble model extends existing ensemble models to ordinal ranking and inherits many useful theoretical properties from them. Next, we discuss one such property: the large-margin bounds.

[4] To precisely replicate the PRank algorithm, the $(K–1)$ binary examples sprouted from a same ordinal example should be considered altogether in updating the perceptron weight vector.

We first list the definition of the margins for a thresholded ensemble (Lin and Li, 2006). Intuitively, we expect the potential value $H_T(\mathbf{x})$ to be in the desired interval (θ_{y-1}, θ_y), and we want $H_T(\mathbf{x})$ to be far from the thresholds.

Definition 2. *Consider a given thresholded ensemble* r *in* (12)*. The normalized margin* $\hat{\rho}_k(\mathbf{x}, y)$ *is defined as*

$$\hat{\rho}_k(\mathbf{x}, y) = \left(2\left[\!\left[k < y\right]\!\right] - 1\right)\left(H_T(\mathbf{x}) - \theta_k\right) \Bigg/ \left(\sum_{t=1}^{T} |\alpha_t| + \sum_{k=1}^{K-1} |\theta_k|\right).$$

Definition 2 is similar to the definition of the support vector machine (SVM) margin made by Shashua and Levin (2003) and is analogous to the definition of the ℓ_1-margins in binary classification (Schapire et al., 1998). A non-positive $\hat{\rho}_k(\mathbf{x}, y)$ indicates an incorrect prediction. We shall now define the Δ-absolute margin cost as

$$E_{\text{in}}(r, \Delta) \;\equiv\; \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K-1}\left[\!\left[\hat{\rho}_k(\mathbf{x}_n, y_n) \leq \Delta\right]\!\right].$$

Consider an ordinal ranking problem such that $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$ only generates examples with the absolute cost vectors. The associated binary classification problem would then be based on an underlying probability distribution $\mathcal{P}_b(\mathbf{X}, Y, W)$ that only generates $W = K-1$ (a constant value). Then, we can obtain a large-margin bound of $E(r)$.

Theorem 5 (Large-margin bound for thresholded ensemble rankers)**.** *Consider a negation complete*[5] *set* $\mathcal{H}$*, which contains only binary classifiers* $h\colon \mathcal{X} \to \{-1, 1\}$ *and is of VC-dimension* d*. Assume that* $\delta > 0$*, and* $N > d + K - 1 = d_E$*. Then, for an ordinal ranking problem with the absolute cost vectors, with probability at least* $1 - \delta$ *over the random choice of the training set* S*, every thresholded ensemble ranker defined from* (12) *satisfies the following bound for all* $\Delta > 0$*:*

$$E(r) \leq E_{\text{in}}(r, \Delta) + O\left(\frac{K}{\sqrt{N}}\left(\frac{d_E \log^2(N/d_E)}{\Delta^2} + \log\frac{1}{\delta}\right)^{1/2}\right).$$

Proof. See Appendix A. □

The bound above can be generalized when $\mathcal{H}$ contains confidence functions rather than binary classifiers using another result of Schapire et al. (1998, Theorem 4) instead.

[5] $\mathcal{H}$ is negation complete if and only if $h \in \mathcal{H} \iff (-h) \in \mathcal{H}$, where $(-h)(\mathbf{x}) = -\big(h(\mathbf{x})\big)$ for all x.

The bound motivates us to design the ORBoost-All algorithm (Lin and Li, 2006), which can be viewed as coupling the reduction framework with a variant of the popular AdaBoost algorithm (Schapire et al., 1998; Schapire and Singer, 1999). ORBoost-All provably minimizes the term $E_{\text{in}}(r, \Delta)$ exponentially fast if the underlying base learner is strong enough. The proof can be made by applying the training error theorem of AdaBoost (Schapire et al., 1998, Theorem 5) on S_E, which is another application of the reduction framework.

5.3 SVM for Ordinal Ranking

SVM is a popular binary classification algorithm (Vapnik, 1995; Schölkopf and Smola, 2002). It maps the feature vector $\mathbf{x}$ to $\phi(\mathbf{x})$ in a possibly higher dimensional space and implicitly computes the inner products with a kernel function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$.

Using a similar set of notations for perceptions (Subsection 5.1), we denote the parallel hyperplanes in the higher dimensional space by $(\mathbf{v}, -\boldsymbol{\theta})$ with an additional bias term b. Now, if we encode $(\mathbf{x}, k)$ with $\mathbf{M} = \gamma \cdot \mathbf{I}_{K-1}$, we can then compute the inner products of the extended examples $(\mathbf{X}^{(k)}, Y^{(k)})$ by

$$\mathcal{K}_E\big((\mathbf{x}, k), (\mathbf{x}', k')\big) = \langle (\phi(\mathbf{x}), \gamma \mathbf{1}_k), (\phi(\mathbf{x}'), \gamma \mathbf{1}_{k'}) \rangle = \mathcal{K}(\mathbf{x}, \mathbf{x}') + \gamma^2 [\![k = k']\!] .$$

With the reduction framework, we can plug in $\mathcal{K}_E$ and $O\big(NK\big)$ extended training examples into the standard SVM to obtain a hyperplane ranker

$$r(\mathbf{x}) = 1 + \sum_{k=1}^{K-1} [\![\langle \mathbf{v}, \phi(\mathbf{x}) \rangle + b - \theta_k > 0]\!] ,$$

based on an optimal solution to

$$\begin{aligned} \min_{\mathbf{v}, b, \theta_k, \xi_n^{(k)}} \quad & \frac{1}{2} \langle \mathbf{v}, \mathbf{v} \rangle + \frac{1}{2\gamma^2} \langle \boldsymbol{\theta}, \boldsymbol{\theta} \rangle + \kappa \sum_{n=1}^{N} \sum_{k=1}^{K-1} W_n^{(k)} \xi_n^{(k)} , \\ \text{subject to} \quad & Y_n^{(k)} \left(\langle \mathbf{v}, \phi(\mathbf{x}) \rangle + b - \theta_k \right) \geq 1 - \xi_n^{(k)}, \\ & \xi_n^{(k)} \geq 0, \text{ for } n = 1, \ldots, N, \text{ and } k = 1, \ldots, K-1. \end{aligned} \tag{13}$$

If $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_{K-1}$, or if the cost vectors considered are convex, Theorems 2 and 3 can guarantee the expected out-of-sample cost of $r(\mathbf{x})$ based on the expected out-of-sample cost of the binary classifier

$$g(\mathbf{x}, k) = \text{sign}\Big(\langle \mathbf{v}, \phi(\mathbf{x}) \rangle + b - \theta_k \Big).$$

The oSVM approach of Cardoso and da Costa (2007) is an instance of (13) with the absolute cost vectors, in which all $W_n^{(k)}$ are equal. The SVOR-IMC approach of Chu and Keerthi (2007) can also be thought as a modified instance of the formulation with the absolute cost vectors, except that the $\frac{1}{2\gamma^2}\langle\theta,\theta\rangle$ term is dropped. Their SVOR-EXC approach is another modified instance using the classification cost vectors plus an additional constraint to guarantee that $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_{K-1}$.

Our proposed algorithm, Reduction-to-SVM (RED-SVM) unifies the above algorithms under a generic formulation (13) with the cost-sensitive reduction framework. RED-SVM can deal with any convex cost vectors by changing $W_n^{(k)}$ and feeding the weighted binary examples to a standard SVM solver, regardless of whether $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_{K-1}$. Interestingly, our earlier work (Li and Lin, 2007b) proved that the ordering property always holds at the optimal SVM solution.

On the other hand, if the cost vectors are ordinal but not convex, solving (13) is more complicated. We adopt a coordinate-descent procedure that switches between optimizing $(\mathbf{v}, b)$ (using the standard SVM solver) and optimizing $\boldsymbol{\theta}$ under the constraints (a small quadratic programming problem with an analytic solution) in the experiments.

Chu and Keerthi (2007) empirically found that SVOR-EXC performed better in terms of the classification cost, and SVOR-IMC preceded in terms of the absolute cost. They explain so by noting that SVOR-EXC minimizes an in-sample loss function that upper-bounds the classification cost, while SVOR-IMC minimizes a loss function that upper-bounds the absolute cost. The explanation is echoed by the study of loss functions for ordinal ranking (Rennie and Srebro, 2005; Dembczyński et al., 2008). The proposed reduction framework offers a more direct explanation than the loss-based one: Because the binary SVM is designed to target for decent out-of-sample binary classification error, reduction with the classification cost (SVOR-EXC) targets for decent out-of-sample classification cost and reduction with the absolute cost (SVOR-IMC) targets for decent out-of-sample absolute cost.

Note that Chu and Keerthi (2007) spent lots of efforts in designing and implementing suitable optimizers for the modified formulation that does not contain the $\frac{1}{2\gamma^2}\langle\theta,\theta\rangle$ term. If we use the standard soft-margin SVM instead, when considering the convex cost vectors like the absolute cost, we can directly and efficiently use the state-of-the-art SVM software to deal with the ordinal ranking problem. The formulation of Chu and

Keerthi (2007) can be approximated by using a large γ. As we shall see in Section 6, even a simple assignment of $\gamma = 1$ performs similarly to the approaches of Chu and Keerthi (2007) in practice.

In addition to the algorithmic benefits described above, the reduction framework can also be used theoretically for SVM. For instance, we demonstrated how we can derive a novel large-margin absolute-cost bound of thresholded ensemble rankers in Subsection 5.2. Next, we extend the bounds to SVM-based formulations and to a wider class of cost functions. While Shashua and Levin (2003) derived one such bound with a specific cost function, their bound is not data dependent and hence does not fully explain the out-of-sample performance of SVM-based rankers in reality (Bartlett and Shawe-Taylor, 1998). Our bound, on the other hand, is not only more general, but also data dependent.

Theorem 6 (Large-margin bound for SVM-based rankers)**.** *Consider a collection*

$$\mathcal{F} = \Big\{ f\big(\mathbf{x}, k\big) = \langle \mathbf{v}, \phi(\mathbf{x}) \rangle + b - \theta_k \colon \|\mathbf{v}\|^2 + \|b - \theta\|^2 \leq 1, \|\phi(\mathbf{x})\|^2 + 1 \leq R^2 \Big\}.$$

Let $B_{\max} = \max_{\mathbf{c} \in \mathcal{C}} \left(\mathbf{c}[1] + \mathbf{c}[K] \right)$*,* $B_{\min} = \min_{\mathbf{c} \in \mathcal{C}} \left(\mathbf{c}[1] + \mathbf{c}[K] \right)$*, and* $\beta = B_{\max} / B_{\min}$*. If* $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_{K-1}$*, or if every* $\mathbf{c}$ *is convex, for any* $\Delta > 0$*, with probability at least* $1 - \delta$*, and for every* f *in* $\mathcal{F}$*, the associated ranker* $r_g(\mathbf{x})$ *with* $g(\cdot) = sign\left(f(\cdot)\right)$ *satisfies*

$$E(r_g) \leq \frac{\beta}{N \cdot (K-1)} \sum_{n=1}^{N} \sum_{k=1}^{K-1} W_n^{(k)} \left[\!\!\left[Y_n^{(k)} f\left(\mathbf{X}_n^{(k)}\right) \leq \Delta \right]\!\!\right] + O\left(\frac{\log N}{\sqrt{N}}, \frac{R}{\Delta}, \sqrt{\log \frac{1}{\delta}} \right).$$

Proof. See Appendix B. □

Thus, if the binary classifier g achieves large margins ($\geq \Delta$) on most of the extended training examples $\left(\mathbf{X}_n^{(k)}, Y_n^{(k)}, W_n^{(k)} \right)$, $E(r_g)$ is guaranteed to be small.

Theorem 6, which is based on the proposed reduction framework, is quite general and applies to a wide class of cost functions. In the special case of the absolute cost function (which results in $W_n^{(k)} = 1$ and $\beta = 1$), Theorem 6 can be simplified to an order-wise comparable bound that has been independently derived by Agarwal (2008) using a similar proving technique.

Note that we can also choose to encode $(\mathbf{x}, k)$ differently. For instance, define

$$\bar{\mathcal{K}}_E\big((\mathbf{x}, k), (\mathbf{x}', k')\big) = \left[\!\!\left[k = k' \right]\!\!\right] \langle \phi_k(\mathbf{x}), \phi_k(\mathbf{x}') \rangle = \left[\!\!\left[k = k' \right]\!\!\right] \mathcal{K}_k(\mathbf{x}, \mathbf{x}').$$

That is, different kernels can be used for different binary classification sub-problems. Recently, Chang et al. (2011) explored such a possibility and proposed the Ordinal Hyperplanes Ranker that achieved promising performance on the age-estimation application. The Ordinal Hyperplanes Ranker can be theoretically justified through the reduction framework using the choice of encoding above. The promising performance suggests the possibility of application opportunities within the proposed general framework.

5.4 Summary

In the previous subsections, we have briefly introduced several existing ordinal ranking algorithms that can be explained as special instances of the reduction framework. We have also derived new cost bounds of the ordinal ranking algorithms via reduction. There are some other existing algorithms that can be viewed as special instances of the reduction framework, as listed in Table 3.

Note that the thresholded linear model is commonly used in statistics for ordinal ranking (Agresti, 2002) and is called the cumulative link model (CLM), which assumes $(\langle \mathbf{v}, \mathbf{x} \rangle - \theta_k)$ to link to the cumulative probability $\mathcal{P}(y \geq k|\mathbf{x})$. CLM can then be coupled with some more assumption on the underlying probability distribution to reach a maximum likelihood solution. The proposed framework treats the thresholded linear model (CLM) as a rank-monotonic special case of the general prediction rule (3). CLM and the proposed framework take very different views on modeling the ordinal ranking problem and hence reach different results. In particular, CLM focuses on deriving from the assumed underlying distribution appropriately, while the proposed framework focuses on using the given cost vectors appropriately.

6 Experiments

We validate the proposed reduction framework by performing experiments with eight benchmark ordinal ranking data sets (Chu and Keerthi, 2007): pyrimdines, machine, boston, abalone, bank, computer, california, census. The data sets were constructed by quantizing some metric regression data sets with $K = 10$. We use the same

Table 3: instances of the reduction framework

ordinal ranking	cost	binary classification algorithm
PRank (Crammer and Singer, 2005)	absolute	modified perceptron rule
kernel ranking (Rajaram et al., 2003)	classification	modified hard-margin SVM
SVOR-EXC SVOR-IMC (Chu and Keerthi, 2007)	classification absolute	modified soft-margin SVM
ORBoost-LR ORBoost-All (Lin and Li, 2006)	classification absolute	modified AdaBoost
oSVM oNN (Cardoso and da Costa, 2007)	absolute absolute	standard soft-margin SVM standard neural network
RED-C4.5 RED-AdaBoost RED-SVM RED-SVM (Li and Lin, 2007b; Lin, 2008)	*any convex* *any convex* *any convex* *any V-shaped*	standard C4.5 standard AdaBoost standard soft-margin SVM modified soft-margin SVM
AdaBoost.OR (Lin and Li, 2009)	*any V-shaped*	standard AdaBoost coupled with special base learners
CLM (Agresti, 2002)	implicitly depends on assumed distribution	maximum likelihood on assumed distribution

training/test ratio and also average the results over 20 trials. Thus, we can fairly compare our results with those of SVOR-IMC and SVOR-EXC (Chu and Keerthi, 2007), the state-of-the-art algorithms.

Table 4: test absolute cost of ordinal ranking algorithms

data	reduction to			SVOR-IMC
set	C4.5	AdaBoost-St	SVM-Perc	Gaussian
pyr.	1.565 ± 0.072	1.360 ± 0.054	$\mathbf{1.304 \pm 0.040}$	$\mathbf{1.294 \pm 0.046}$
mac.	$0.987 \pm 0.024^*$	$0.875 \pm 0.017^*$	$\mathbf{0.842 \pm 0.022}^*$	0.990 ± 0.026
bos.	0.950 ± 0.016	0.846 ± 0.015	$\mathbf{0.732 \pm 0.013}^*$	0.747 ± 0.011
aba.	1.560 ± 0.006	1.458 ± 0.005	1.383 ± 0.004	$\mathbf{1.361 \pm 0.003}$
ban.	1.700 ± 0.005	1.481 ± 0.002	1.404 ± 0.002	$\mathbf{1.393 \pm 0.002}$
com.	0.701 ± 0.003	0.604 ± 0.002	$\mathbf{0.565 \pm 0.002}^*$	0.596 ± 0.002
cal.	$0.974 \pm 0.004^*$	$0.991 \pm 0.003^*$	$\mathbf{0.940 \pm 0.001}^*$	1.008 ± 0.001
cen.	1.263 ± 0.003	1.210 ± 0.001	$\mathbf{1.143 \pm 0.002}^*$	1.205 ± 0.002

(those within one standard error of the lowest one are marked in bold)

(those better than SVOR-IMC are marked with *)

6.1 The Absolute Cost

We first test the reduction framework with the absolute cost vectors, $\mathbf{M} = \gamma \cdot \mathbf{I}_{K-1}$ with $\gamma = 1$, and three different binary classification algorithms. The first binary algorithm is the C4.5 decision tree (Quinlan, 1986).[6] The second is AdaBoost-St, which uses AdaBoost (Schapire et al., 1998) to aggregate 500 decision stumps. The third one is SVM-Perc, which is SVM (Vapnik, 1995) with the perceptron kernel (Lin and Li, 2008). The parameter κ of the soft-margin SVM is determined by a 5-fold cross validation procedure with $\log_2 \kappa \in \{-17, -15, \ldots, 3\}$ (Hsu et al., 2003), and LIBSVM (Chang and Lin, 2001) is adopted as the solver.

We list the mean and the standard error of the test absolute costs in Table 4, with entries within one standard error of the lowest one marked in bold.[7] With the proposed reduction framework, all the three binary learning algorithms, even the simplest C4.5 decision tree, could be better than SVOR-IMC with the Gaussian kernel on some of the data sets. The results demonstrate that all the algorithms can achieve decent out-of-

[6] C4.5 can directly take the extended input vector $(\mathbf{x}, k)$ without encoding. We choose to still encode $(\mathbf{x}, k)$ by the matrix $\mathbf{M} = \gamma \cdot \mathbf{I}_{K-1}$ to make a simple and fair comparison with the other two algorithms that need the encoding.

[7] Note that the results from Chu and Keerthi (2007) include the standard deviation and here we compute the standard error instead.

sample performances. Among the three algorithms, reduction to SVM-Perc is usually better than the other two.

Note, however, that Chu and Keerthi (2007) use the Gaussian kernel rather than the perceptron kernel in their experiments. For a fair comparison, we implement SVOR-IMC with the perceptron kernel by modifying LIBSVM (Chang and Lin, 2001) and conduct experiments with the parameter selection procedure introduced earlier in this section. We also couple RED-SVM with the Gaussian kernel and the parameter selection procedure of SVOR-IMC (Chu and Keerthi, 2007).

In addition, to examine the performance of different SVM-based approaches on real-world ordinal ranking problems, we include two more data sets: car and the red wine subset (redwine) of the wine quality set from the UCI machine learning repository (Hettich et al., 1998). The car problem aims at ranking cars to four conditions: $\{\texttt{unacceptable}, \texttt{acceptable}, \texttt{good}, \texttt{very good}\}$; the redwine problem ranks red wine samples to 11 different levels between 0 and 10, while the actual data only contain samples with ranks between 3 and 8. We randomly split 75% of the examples for training and 25% for testing, and conduct 20 runs of such a random split. The training input vectors are first scaled to $[0, 1]$ linearly, and the test input vectors are scaled accordingly.

Table 5 list the results, which suggest that direct reduction to the standard SVM (RED-SVM) performs similarly to SVOR-IMC when using the same kernel. RED-SVM, nevertheless, is much easier to implement. In addition, RED-SVM is significantly faster than SVOR-IMC in training. The speed difference is illustrated in Figure 2 using the four largest data sets. We make a fair comparison by implementing both algorithms under the same code/data structure of LIBSVM. The CPU time was gathered on a 1.7G Dual Intel Xeon machine with 1GB RAM. After a careful comparison, we find that the main cause to the time difference is the speed-up heuristics. While, to the best of our knowledge, not much has been done to improve the original SVOR-IMC algorithm, plenty of heuristics, such as shrinking and advanced working selection in LIBSVM, can be seamlessly adopted by RED-SVM because of the reduction framework. The newly-designed SVOR-IMC does not enjoy the same advantage. The difference demonstrate an important property of the reduction framework: Any improvements to the binary classification approaches can be immediately inherited by reduction-based

Table 5: test absolute cost of SVM-based ordinal ranking algorithms

data set	RED-SVM perceptron	RED-SVM Gaussian	SVOR-IMC perceptron	SVOR-IMC Gaussian
pyr.	**1.304±0.040**	**1.277±0.037**	1.315±0.039	**1.294±0.046**
mac.	0.842±0.022	0.914±0.026	**0.814±0.019**	0.990±0.026
bos.	**0.732±0.013**	0.752±0.015	**0.729±0.013**	0.747±0.011
aba.	1.383±0.004	**1.361±0.003**	1.386±0.005	**1.361±0.003**
ban.	1.404±0.002	1.395±0.002	1.404±0.002	**1.393±0.002**
com.	**0.565±0.002**	0.588±0.001	**0.565±0.002**	0.596±0.002
cal.	**0.940±0.001**	0.945±0.001	**0.939±0.001**	1.008±0.001
cen.	**1.143±0.002**	1.167±0.002	**1.143±0.002**	1.205±0.002
car	0.061±0.003	**0.050±0.002**	0.064±0.003	**0.051±0.002**
red.	**0.357±0.005**	0.425±0.004	**0.357±0.005**	0.429±0.004

(those within one standard error of the lowest one are marked in bold)

ordinal ranking algorithms.

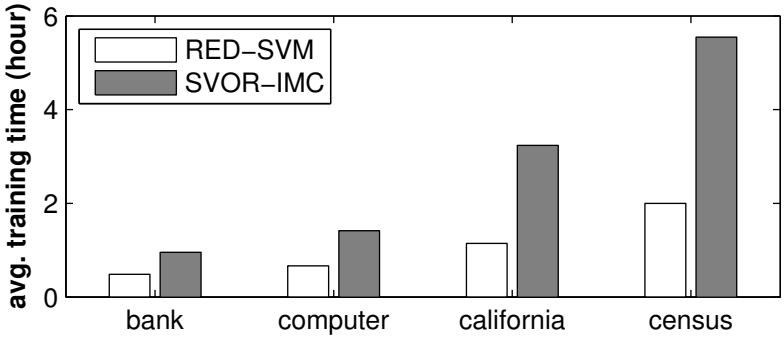

Figure 2: training time (including automatic parameter selection) of SVM-based ordinal ranking algorithms with the perceptron kernel

6.2 The Classification Cost

We also test the reduction framework with the classification cost vectors. Because the classification cost vectors are V-shaped but not convex, the reduction framework only guarantees to work when the obtained binary classifier is rank-monotonic. Such a condition is not easily met by C4.5 nor AdaBoost. Thus, we only test the reduction framework using a variant of RED-SVM that respects the constraint $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_{K-1}$

Table 6: test classification cost of SVM-based ordinal ranking algorithms

data set	RED-SVM perceptron	RED-SVM Gaussian	SVOR-EXC Gaussian
pyr.	$\mathbf{0.762 \pm 0.021}$	0.787 ± 0.021	$\mathbf{0.752 \pm 0.014}$
mac.	$\mathbf{0.572 \pm 0.013}$	0.637 ± 0.016	0.661 ± 0.012
bos.	$\mathbf{0.541 \pm 0.009}$	0.565 ± 0.008	0.569 ± 0.006
aba.	0.721 ± 0.002	$\mathbf{0.708 \pm 0.002}$	0.736 ± 0.002
ban.	0.751 ± 0.001	0.746 ± 0.001	$\mathbf{0.744 \pm 0.001}$
com.	$\mathbf{0.451 \pm 0.002}$	0.461 ± 0.001	0.462 ± 0.001
cal.	$\mathbf{0.613 \pm 0.001}$	$\mathbf{0.612 \pm 0.001}$	0.640 ± 0.001
cen.	0.688 ± 0.001	$\mathbf{0.686 \pm 0.001}$	0.699 ± 0.000
car	0.064 ± 0.003	$\mathbf{0.050 \pm 0.002}$	0.054 ± 0.002
red.	$\mathbf{0.327 \pm 0.005}$	0.392 ± 0.004	0.403 ± 0.004

(those within one standard error of the lowest one are marked in bold)

(see Subsection 5.3), and compare the variant with SVOR-EXC.

We list the mean and the standard error of the test classification costs in Table 6, with entries within one standard error of the lowest one marked in bold. RED-SVM with the perceptron kernel is better than RED-SVM with the Gaussian kernel on most of the benchmark data sets and redwine, while RED-SVM with the Gaussian kernel is better on car. RED-SVM with the Gaussian kernel is in term slightly better than SVOR-EXC with the Gaussian kernel on most of the data sets. The results again justify the usefulness of the proposed reduction framework.[8]

6.3 Other Costs

Next, we use different cost vectors for evaluation to demonstrate the power of the proposed cost-sensitive ordinal ranking framework. We consider two different kinds of cost vectors. First, we define the asymmetric cost vector for rank ℓ as

$$\mathbf{c}^{(\ell)}[k] = \begin{cases} 2^{k-\ell} & \text{if } (\ell - \frac{K+1}{2})(\ell - k) > 0, \\ \frac{1}{2}|k - \ell| & \text{otherwise.} \end{cases}$$

[8] We do not have the results of SVOR-EXC with the perceptron kernel because it is difficult to use LIBSVM to implement and compare SVOR-EXC fairly with RED-SVM.

That is, for $K = 10$, an asymmetric cost vector for $(\mathbf{x}, 3)$ would be

$$\mathbf{c}^{(3)} = [1, 0.5, 0, 1, 2, 4, 8, 16, 32, 64].$$

The asymmetric cost vector combines two different cost vectors. For instance, when $\ell < \frac{K+1}{2}$, the asymmetric cost vector includes a fast-growing cost vector when $k > \ell$ and a slow-growing one when $k \leq \ell$. A potential use of the asymmetric cost vector is to tolerate the cases when k is on the "same side" of ℓ while penalizing the cases when k is far from ℓ.

Another cost vector that we consider is called two-Gaussian (2Gauss), which combines two (reverted) Gaussian functions. The formal definition of the 2Gauss cost is

$$\mathbf{c}^{(\ell)}[k] = \left(1 - \exp\left(-\frac{1}{8}(k-\ell)^2\right)\right) \cdot \begin{cases} 5 & \text{if } (\ell - \frac{K+1}{2})(\ell - k) > 0, \\ 1 & \text{otherwise.} \end{cases}$$

Note that the 2Gauss cost vectors are V-shaped but not convex. They also penalize the two sides of cases differently.

Table 7 lists the mean and standard error of the test asymmetric costs. For RED-SVM, we consider three different kinds of costs for training: asymmetric, absolute and classification. We then compare the three variants of RED-SVM with the state-of-the-art SVOR-IMC and SVOR-EXC. First of all, RED-SVM with the asymmetric cost and RED-SVM with the absolute cost generally perform better than SVOR-IMC or SVOR-EXC, which demonstrates that the proposed cost-sensitive framework could achieve decent test performance in a cost-sensitive setting.

The classification cost vectors are very different from the asymmetric ones and thus RED-SVM with the classification cost cannot perform well when evaluated with the asymmetric cost vectors. The absolute cost vectors, on the other hand, are closer to the asymmetric ones. In fact, Table 7 suggests that RED-SVM with the absolute cost is often better than RED-SVM with the asymmetric cost. Thus, when evaluating with convex cost vectors like the asymmetric ones, training with the absolute cost vectors could be a useful first-hand choice.

Table 8 lists the mean and standard error of the test 2Gauss costs. For RED-SVM, we also consider three different kinds of costs during training: 2Gauss, absolute and classification. Note that RED-SVM with the 2Gauss cost is not only better than the

Table 7: test asymmetric cost of SVM-based ordinal ranking algorithms

data	RED-SVM			SVOR-IMC	SVOR-EXC
set	asymmetric	absolute	classification		
pyr.	1.716±0.182	**1.593±0.118**	4.522±1.505	**1.665±0.140**	2.309±0.321
mac.	0.873±0.056	**0.820±0.034**	**0.814±0.051**	0.898±0.046	1.011±0.062
bos.	**0.762±0.038**	**0.759±0.030**	**0.750±0.029**	0.784±0.049	0.822±0.063
aba.	1.992±0.022	1.995±0.018	2.700±0.035	**1.952±0.015**	2.580±0.024
ban.	1.937±0.009	**1.923±0.009**	2.558±0.032	1.948±0.010	2.490±0.013
com.	**0.492±0.003**	0.508±0.002	0.507±0.003	0.533±0.002	0.535±0.003
cal.	1.183±0.007	**1.141±0.005**	1.223±0.008	1.208±0.007	1.318±0.009
cen.	1.587±0.010	**1.552±0.007**	1.778±0.019	1.746±0.019	2.023±0.021

(those within one standard error of the lowest one are marked in bold)

state-of-the-art SVOR-IMC and SVOR-EXC, but also better than other RED-SVM variants. One possible explanation is that RED-SVM with the absolute cost cannot perform well because the absolute cost is convex while 2Gauss is not; RED-SVM with the classification cost also cannot perform well because the classification cost is symmetric on both sides of the desired label ℓ while 2Gauss is not. The results justify the importance of the proposed cost-sensitive ordinal ranking framework.

6.4 Improving NDCG with Cost-sensitive Ordinal Ranking

We demonstrate another usefulness of cost-sensitive ordinal ranking by designing a cost vector that could help improve the Normalized Discounted Cumulative Gain (NDCG), a criteria commonly used in listwise ranking (Liu, 2009). The design uses a bound from the McRank work of Li et al. (2008), who showed that for any set of test input vectors $\{\mathbf{x}'_m\}_{m=1}^M$ with ideal ranks y_m,

$$1 - \text{NDCG} \leq const \cdot \sqrt{\sum_{m=1}^{M} \mathbf{c}^{(y_m)}[r(\mathbf{x}'_m)]}, \text{ where } \mathbf{c}^{(\ell)}[k] = \left(2^\ell - 2^k\right)^2. \tag{14}$$

Nevertheless, the original McRank algorithm was not designed with the bound above, but was derived by replacing $\mathbf{c}$ with $(2^K - 1)^2$ times the classification cost—a much looser upper bound. Next, we examine whether we can use the tighter cost vector in (14) to achieve better (higher) NDCG performance.

Table 8: test 2Gauss cost of SVM-based ordinal ranking algorithms

data	RED-SVM			SVOR-IMC	SVOR-EXC
set	2Gauss	absolute	classification		
pyr.	**0.760±0.055**	0.961±0.040	1.025±0.071	0.930±0.047	0.932±0.047
mac.	**0.456±0.019**	0.505±0.019	**0.466±0.020**	0.552±0.024	0.540±0.030
bos.	**0.383±0.015**	0.434±0.013	0.435±0.013	0.434±0.014	0.425±0.012
aba.	**0.935±0.006**	1.032±0.004	0.943±0.005	1.006±0.004	**0.936±0.005**
ban.	**0.912±0.003**	1.069±0.003	1.015±0.004	1.051±0.003	0.997±0.003
com.	**0.227±0.001**	0.279±0.002	0.263±0.002	0.287±0.001	0.274±0.002
cal.	**0.542±0.002**	0.607±0.001	0.577±0.003	0.602±0.002	0.578±0.002
cen.	**0.713±0.001**	0.786±0.002	0.737±0.002	0.799±0.002	0.765±0.003

(those within one standard error of the lowest one are marked in bold)

We transform the benchmark data sets to listwise ranking by randomly generating 10000 subsets of size 10. Then, we evaluate NDCG at the 10-th position for each subset and report the average. Note that the original McRank algorithm (Li et al., 2008) is very similar to reduction with the absolute cost, with a possibly weaker underlying learner (boosting tree) and slightly different rule of converting g to r_g.[9] Thus, in addition to the ndcg cost (14), we also couple the absolute cost (similar to the actual McRank algorithm) and the classification cost (similar to the theoretical backbone of McRank) with RED-SVM for comparison. We then compare the three variants with the state-of-the-art SVOR-IMC and SVOR-EXC. Table 9 lists the mean and standard error of the test NDCG on the eight data sets. We see that RED-SVM with the ndcg cost often achieves better NDCG performance than the other two variants of RED-SVM, including RED-SVM with the classification cost. Also, RED-SVM with the ndcg cost is also often better than SVOR-IMC and SVOR-EXC. The results demonstrate the potential of cost-sensitive ordinal ranking on improving listwise ranking, which echoes the recent finding of a related work (Tsai et al., 2010) towards the Yahoo! Learning to Rank Challenge.

[9] Although McRank is designed from the classification cost, a closer inspection from the reduction perspective reveals that the algorithm can be better interpreted by the absolute cost.

Table 9: test NDCG of ordinal ranking algorithms

data	RED-SVM			SVOR-IMC	SVOR-EXC
set	ndcg	absolute	classification		
pyr.	0.924±0.008	**0.934±0.008**	0.917±0.011	**0.933±0.008**	0.922±0.010
mac.	0.973±0.002	0.973±0.002	**0.976±0.001**	0.961±0.003	0.956±0.004
bos.	**0.958±0.002**	**0.957±0.002**	**0.957±0.002**	**0.956±0.003**	0.953±0.003
aba.	**0.872±0.001**	0.865±0.001	0.869±0.002	0.868±0.001	0.864±0.001
ban.	**0.902±0.000**	0.879±0.000	0.881±0.001	0.879±0.000	0.880±0.001
com.	0.961±0.000	0.961±0.000	**0.962±0.000**	0.959±0.000	0.959±0.000
cal.	**0.934±0.000**	0.931±0.000	0.932±0.000	0.930±0.000	0.931±0.000
cen.	**0.929±0.000**	0.919±0.000	0.922±0.001	0.914±0.000	0.917±0.001

(those within one standard error of the highest one are marked in bold)

7 Conclusion

We presented the reduction framework from ordinal ranking to binary classification. The framework is accompanied by the flexibility to work with any reasonable cost vectors and any binary classifiers. We showed the theoretical guarantees of the framework, including the cost bound, the regret bound, and the equivalence between ordinal ranking and binary classification.

We also demonstrated the advantages of the framework in designing new algorithms, explaining existing ones, and deriving new generalization bounds for ordinal ranking. Furthermore, the usefulness of the framework was empirically validated by comparing the newly-proposed algorithms constructed from the framework with the state-of-the-art SVOR-IMC and SVOR-EXC algorithms. In particular, the proposed cost-sensitive ordinal ranking algorithms were observed to not only improve over SVOR-IMC and SVOR-EXC when using common evaluation criteria like the absolute or the classification costs, but also superior over SVOR-IMC and SVOR-EXC when evaluated with other costs as well as the NDCG criteria for listwise ranking.

Acknowledgments

We wish to thank Yaser S. Abu-Mostafa, Amrit Pratap, John Langford and the anonymous reviewers for valuable discussions and comments. When this project was initiated, Ling Li was supported by the Caltech SISL Graduate Fellowship, and Hsuan-Tien Lin was supported by the Caltech EAS Division Fellowship. The continuing work was supported by the National Science Council of Taiwan via the grant NSC 98-2221-E-002-192.

References

Abe, N., Zadrozny, B., and Langford, J. (2004). An iterative method for multi-class cost-sensitive learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3–11.

Agarwal, S. (2008). Generalization bounds for some ordinal regression algorithms. In *Algorithmic Learning Theory*, pages 7–21.

Agresti, A. (2002). *Categorical Data Analysis*. Wiley, second edition.

Ailon, N. and Mohri, M. (2008). An efficient reduction of ranking to classification. In *Learning Theory: 21st Annual Conference on Learning Theory*, pages 87–98.

Anderson, J. A. (1984). Regression and ordered categorical variables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46:1–30.

Balcan, M.-F., Bansal, N., Beygelzimer, A., Coppersmith, D., Langford, J., and Sorkin, G. B. (2007). Robust reductions from ranking to classification. In *Learning Theory: 20th Annual Conference on Learning Theory*, pages 604–619.

Bartlett, P. L. and Shawe-Taylor, J. (1998). Generalization performance of support vector machines and other pattern classifiers. In *Advances in Kernel Methods: Support Vector Learning*, pages 43–54.

Beygelzimer, A., Daniand, V., Hayes, T., Langford, J., and Zadrozny, B. (2005). Error limiting reductions between classification tasks. In *Machine Learning: Proceedings of the 22rd International Conference*, pages 49–56.

Beygelzimer, A., Langford, J., and Ravikumar, P. (2007). Multiclass classification with filter trees. Downloaded from `http://hunch.net/~jl`.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. In *Machine Learning: Proceedings of the 24th International Conference*, pages 129–136.

Cardoso, J. S. and da Costa, J. F. P. (2007). Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research*, 8:1393–1429.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: A Library for Support Vector Machines*. National Taiwan University. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chang, K.-Y., Chen, C.-S., and Hung, Y.-P. (2011). Ordinal hyperplanes ranker with cost sensitivities for age estimation. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, pages 585–592.

Chu, W. and Ghahramani, Z. (2005). Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041.

Chu, W. and Keerthi, S. S. (2007). Support vector ordinal regression. *Neural Computation*, 19:792–815.

Cossock, D. and Zhang, T. (2008). Statistical analysis of Bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54:4140–5154.

Crammer, K. and Singer, Y. (2005). Online ranking by projecting. *Neural Computation*, 17:145–175.

Dembczyński, K., Kotłowski, W., and Słowiński, R. (2008). Ordinal classification with decision rules. In *Proceedings of the 3rd International Workshop on Mining Complex Data*, pages 169–181.

Figueira, J., Greco, S., and Ehrgott, M., editors (2005). *Multiple Criteria Decision Analysis: State of the Art Surveys*. Springer-Verlag.

Frank, E. and Hall, M. (2001). A simple approach to ordinal classification. In *Machine Learning: Proceedings of the 12th European Conference on Machine Learning*, pages 145–156.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Greco, S., Słowiński, R., and Matarazzo, B. (2000). Extension of the rough set approach to multicriteria decision support. *European Journal of Operational Research*, 38:161–195.

Har-Peled, S., Roth, D., and Zimak, D. (2003). Constraint classification: A new approach to multiclass classification and ranking. In *Advances in Neural Information Processing Systems: Proceedings of the 2002 Conference*, volume 15, pages 365–379.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132.

Hettich, S., Blake, C. L., and Merz, C. J. (1998). UCI repository of machine learning databases. Downloadable at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. Technical report, National Taiwan University.

Joachims, T. (2006). Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226.

Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Kotłowski, W. and Słowiński, R. (2009). Rule learning with monotonicity constraints. In *Machine Learning: Proceedings of the 26th International Conference*, pages 537–544.

Li, L. and Lin, H.-T. (2007a). Optimizing 0/1 loss for perceptrons by random coordinate descent. In *Proceedings of the 2007 International Joint Conference on Neural Networks*, pages 749–754.

Li, L. and Lin, H.-T. (2007b). Ordinal regression by extended binary classification. In *Advances in Neural Information Processing Systems: Proceedings of the 2006 Conference*, volume 19, pages 865–872.

Li, P., Burges, C., and Wu, Q. (2008). McRank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems: Proceedings of the 2007 Conference*, volume 20, pages 897–904.

Lin, H.-T. (2008). *From Ordinal Ranking to Binary Classification*. PhD thesis, California Institute of Technology.

Lin, H.-T. and Li, L. (2006). Large-margin thresholded ensembles for ordinal regression: Theory and practice. In *Algorithmic Learning Theory*, pages 319–333.

Lin, H.-T. and Li, L. (2008). Support vector machinery for infinite ensemble learning. *Journal of Machine Learning Research*, 9:285–312.

Lin, H.-T. and Li, L. (2009). Combining ordinal preferences by boosting. In *Preference Learning Workshop at ECML/PKDD*.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42:109–142.

Meir, R. and Rätsch, G. (2003). An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning*, pages 119–184.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Rajaram, S., Garg, A., Zhou, X. S., and Huang, T. S. (2003). Classification approach towards ranking and sorting problems. In *Machine Learning: Proceedings of the 14th European Conference on Machine Learning*, pages 301–312.

Rennie, J. D. M. and Srebro, N. (2005). Loss functions for preference levels: Regression with discrete ordered labels. In *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, pages 180–186.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books.

Schapire, R. E., Freund, Y., Bartlett, P. L., and Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms: Using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels*. MIT Press.

Shashua, A. and Levin, A. (2003). Ranking with large margin principle: Two approaches. In *Advances in Neural Information Processing Systems: Proceedings of the 2002 Conference*, volume 15, pages 937–944.

Sill, J. (1998). Monotonic networks. In *Advances in Neural Information Processing Systems: Proceedings of the 1997 Conference*, volume 10, pages 661–667.

Słowiński, R., Greco, S., and Matarazzo, B. (2007). Dominance-based rough set approach to reasoning about ordinal data. In *Proceedings of the international conference on Rough Sets and Intelligent Systems Paradigms*, pages 5–11.

Tsai, M.-F., Chen, S.-T., Chen, Y.-N., Ferng, C.-S., Wang, C.-H., Wen, T.-Y., and Lin, H.-T. (2010). An ensemble ranking solution to the yahoo! learning to rank challenge. Technical report, National Taiwan University.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Xia, F., Zhou, L., Yang, Y., and Zhang, W. (2007). Ordinal regression as multiclass classification. *International Journal of Intelligent Control and Systems*, 12(3):230–236.

Zadrozny, B., Langford, J., and Abe, N. (2003). Cost sensitive learning by cost-proportionate example weighting. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 435–442.

A Proof of Theorem 5

Proof. Consider the extended training set

$$S_E = \left\{ \left(\mathbf{X}_n^{(k)}, Y_n^{(k)}, W_n^{(k)} = K-1 \right) : 1 \leq n \leq N, 1 \leq k \leq K-1 \right\}$$

with $N(K-1)$ elements. If we directly draw from $\mathcal{P}_b$, each element is a possible outcome. Note, however, that these elements are not all independent outcomes. For example, $\left(\mathbf{X}_n^{(1)}, Y_n^{(1)}, W_n^{(1)} \right)$ and $\left(\mathbf{X}_n^{(2)}, Y_n^{(2)}, W_n^{(2)} \right)$ are dependent because they sprout from the same $(\mathbf{x}_n, y_n, \mathbf{c}_n)$. Thus, we cannot directly use the whole S_E as a set of independent outcomes from $\mathcal{P}_b$.

Nevertheless, some subsets of S_E contain independent outcomes from $\mathcal{P}_b$. One way to extract such a subset is to choose one k_n uniformly and independently from $\{1, \ldots, K-1\}$ for each training example $(\mathbf{x}_n, y_n, \mathbf{c}_n)$. The resulting subset would be named

$$S_I = \left\{ \left(\mathbf{X}_n^{(k_n)}, Y_n^{(k_n)}, W_n^{(k_n)} = K-1 \right) \right\}_{n=1}^{N}.$$

We will use a simple encoding scheme of $\mathbf{M} = \mathbf{I}_{K-1}$ to represent $\mathbf{X}^{(k)} = (\mathbf{x}, k)$. Then, consider a binary classification ensemble $g(\mathbf{X}^{(k)})$ defined by a linear combination of the functions in

$$\mathcal{G} = \left\{ \tilde{h} \colon \tilde{h}(\mathbf{X}^{(k)}) = h(\mathbf{x}), h \in \mathcal{H} \right\} \cup \left\{ s_\ell \right\}_{\ell=1}^{K-1}. \tag{15}$$

Here $s_\ell(\mathbf{X}^{(k)})$ is a decision stump on dimension $d+\ell$ (Holte, 1993). If the output space of s_ℓ is $\{-1,1\}$, it is not hard to show that the VC-dimension of $\mathcal{G}$ is no more than $d_E = d + K - 1$. Since the proof of Schapire et al. (1998, Theorem 2), which will be applied on $\mathcal{G}$ later, only requires a combinatorial counting bound on the possible outputs of s_ℓ, we let

$$s_\ell(\mathbf{X}^{(k)}) = -\frac{\text{sign}\Big(\mathbf{X}^{(k)}[d+\ell] - 0.5\Big) + 1}{2} = -[\![k = \ell]\!] \in \{-1, 0\}$$

to get a cosmetically cleaner proof. Some different versions of the bound can be obtained by considering $s_\ell(\mathbf{X}^{(k)}) \in \{-1,1\}$ or by bounding the number of possible outputs of s_ℓ directly by a tighter term.

Without loss of generality, we normalize r such that $\sum_{t=1}^{T} |\alpha_t| + \sum_{\ell=1}^{K-1} |\theta_\ell|$ is 1. Then, consider an ensemble function

$$g(\mathbf{X}^{(k)}) = H_T(\mathbf{x}) - \theta_k = \sum_{t=1}^{T} \alpha_t \tilde{h}_t(\mathbf{X}^{(k)}) + \sum_{k=1}^{K-1} \theta_\ell s_\ell(\mathbf{X}^{(k)}).$$

For every $(\mathbf{X}^{(k)}, Y^{(k)}, W^{(k)})$ derived from the tuple $(\mathbf{x}, y, k)$, the term $\Big(Y^{(k)} \cdot g\left(\mathbf{X}^{(k)}\right)\Big) = \hat{\rho}_k(\mathbf{x}, y)$. Furthermore, we can easily see that $r = r_g$. Thus, by Theorem 2,

$$E(r) = E(r_g) \leq E_b(g). \tag{16}$$

Because S_I contains N independent outcomes from $\mathcal{P}_b(\mathbf{X}, Y, W)$, the large-margin theorem (Schapire et al., 1998, Theorem 2) states that with probability at least $1 - \frac{\delta}{2}$ over the choice of S_I,

$$\begin{aligned} E_b(g) &= \mathop{\mathbb{E}}_{(\mathbf{X},Y,W)\sim\mathcal{P}_b} W \cdot [\![Y \neq g(\mathbf{X})]\!] \\ &\leq \frac{K-1}{N} \sum_{n=1}^{N} [\![Y_n^{(k_n)} \cdot g\left(\mathbf{X}_n^{(k_n)}\right) \leq \Delta]\!] \\ &\quad + O\left(\frac{K}{\sqrt{N}} \left(\frac{d_E \log^2(N/d_E)}{\Delta^2} + \log\frac{1}{\delta}\right)^{1/2}\right). \end{aligned} \tag{17}$$

Define a Boolean random variable

$$b_n \equiv [\![Y_n^{(k_n)} g\left(\mathbf{X}_n^{(k_n)}\right) \leq \Delta]\!] = [\![\hat{\rho}_{k_n}(\mathbf{x}_n, y_n) \leq \Delta]\!].$$

We see that b_n comes with mean $\frac{1}{K-1} \sum_{k=1}^{K-1} [\![\hat{\rho}_k(\mathbf{x}_n, y_n) \leq \Delta]\!]$. Using Hoeffding's inequality (Hoeffding, 1963), when each b_n is chosen independently, with probability at

least $1-\frac{\delta}{2}$ over the choice of b_n,

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N} b_n &\leq \frac{1}{N}\sum_{n=1}^{N}\frac{1}{K-1}\sum_{k=1}^{K-1}[\![\hat{\rho}_k(\mathbf{x}_n, y_n)\leq\Delta]\!] + O\left(\frac{1}{\sqrt{N}}\left(\log\frac{1}{\delta}\right)^{1/2}\right)\\ &= \frac{1}{K-1}E_{\text{in}}(r,\Delta) + O\left(\frac{1}{\sqrt{N}}\left(\log\frac{1}{\delta}\right)^{1/2}\right). \qquad (18)\end{aligned}$$

The desired result can be proved by combining (16), (17), and (18) with a union bound.

□

B Proof of Theorem 6

Proof. For every example $(\mathbf{x}, y, \mathbf{c})$, by the same derivation as Theorem 2,

$$\begin{aligned}&(K-1)\cdot\mathbf{c}[r(\mathbf{x})]\\ \leq\ &\sum_{k=1}^{K-1} W^{(k)}\left[\!\left[Y^{(k)}f\left(\mathbf{X}^{(k)}\right)\leq 0\right]\!\right]\\ \leq\ &(K-1)\cdot(\mathbf{c}[1]+\mathbf{c}[K])\cdot\sum_{k=1}^{K-1}\frac{W^{(k)}}{(K-1)\cdot(\mathbf{c}[1]+\mathbf{c}[K])}\left[\!\left[Y^{(k)}f\left(\mathbf{X}^{(k)}\right)\leq 0\right]\!\right].\end{aligned}$$

Note that

$$P^{(k)} = \frac{W^{(k)}}{(K-1)\cdot(\mathbf{c}[1]+\mathbf{c}[K])}$$

sums to 1. Then, for each example $(\mathbf{x}, y, \mathbf{c})$ obtained from $\mathcal{P}(\mathbf{x}, y, \mathbf{c})$, we can randomly choose k according to $P^{(k)}$ and form an unweighted binary example $\left(\mathbf{X}^{(k)}, Y^{(k)}\right)$. The procedure above defines a probability distribution $\mathcal{P}_u\left(\mathbf{X}^{(k)}, Y^{(k)}\right)$. Integrating over all $(\mathbf{x}, y, \mathbf{c})$, we get

$$E(r_f) \leq B_{\max}\mathop{\mathbb{E}}_{(\mathbf{X}^{(k)},Y^{(k)})\sim\mathcal{P}_u}\left[\!\left[Y^{(k)}f\left(\mathbf{X}^{(k)}\right)\leq 0\right]\!\right].$$

When each k_n is chosen independently according to $P_n^{(k)}$, we can generate N independent examples $\left(\mathbf{X}_n^{(k_n)}, Y_n^{(k_n)}\right)$ from $\mathcal{P}_u\left(\mathbf{X}^{(k)}, Y^{(k)}\right)$ and S. Then, using a cost bound for SVM in binary classification (Bartlett and Shawe-Taylor, 1998), with probability at least $(1-\frac{\delta}{2})$ over the choice of $\left\{\left(\mathbf{X}_n^{(k_n)}, Y_n^{(k_n)}\right)\right\}_{n=1}^{N}$,

$$\begin{aligned}&\mathop{\mathbb{E}}_{(\mathbf{X}^{(k)},Y^{(k)})\sim\mathcal{P}_u}\left[\!\left[Y^{(k)}f\left(\mathbf{X}^{(k)}\right)\leq 0\right]\!\right]\\ \leq\ &\frac{1}{N}\sum_{n=1}^{N}\left[\!\left[Y_n^{(k_n)}f\left(\mathbf{X}_n^{(k_n)}\right)\leq\Delta\right]\!\right] + O\left(\frac{\log N}{\sqrt{N}}, \frac{R}{\Delta}, \sqrt{\log\frac{1}{\delta}}\right).\end{aligned}$$

Using the same technique as the proof of Theorem 5 with $b_n = \left[\!\left[Y_n^{(k_n)} f\left(\mathbf{X}_n^{(k_n)}\right) \leq \Delta \right]\!\right]$ and a union bound, with probability $> 1 - \delta$,

$$
\begin{aligned}
& E(r_f) \\
\leq\ & \frac{B_{\max}}{N} \sum_{n=1}^{N} \left[\!\left[Y_n^{(k_n)} f\left(\mathbf{X}_n^{(k_n)}\right) \leq \Delta \right]\!\right] + O\left(\frac{\log N}{\sqrt{N}}, \frac{R}{\Delta}, \sqrt{\log \frac{1}{\delta}}\right) \\
\leq\ & \frac{B_{\max}}{N} \sum_{n=1}^{N} \frac{1}{(K-1) \cdot (\mathbf{c}_n[1] + \mathbf{c}_n[K])} \sum_{k=1}^{K-1} W_n^{(k)} \cdot \left[\!\left[Y_n^{(k)} f\left(\mathbf{X}_n^{(k)}\right) \leq \Delta \right]\!\right] \\
& + O\left(\frac{\log N}{\sqrt{N}}, \frac{R}{\Delta}, \sqrt{\log \frac{1}{\delta}}\right) + O\left(\frac{1}{\sqrt{N}}, \sqrt{\log \frac{1}{\delta}}\right) \\
\leq\ & \frac{\beta}{N \cdot (K-1)} \sum_{n=1}^{N} \sum_{k=1}^{K-1} W_n^{(k)} \cdot \left[\!\left[Y_n^{(k)} f\left(\mathbf{X}_n^{(k)}\right) \leq \Delta \right]\!\right] + O\left(\frac{\log N}{\sqrt{N}}, \frac{R}{\Delta}, \sqrt{\log \frac{1}{\delta}}\right).
\end{aligned}
$$

□