On Equivalence Relationships Between Classification and Ranking Algorithms

Şeyda Ertekin   SEYDA@MIT.EDU
Cynthia Rudin   RUDIN@MIT.EDU
MIT Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142, USA

Editor: Yoav Freund

Abstract

We demonstrate that there are machine learning algorithms that can achieve success for two separate tasks simultaneously, namely the tasks of classification and bipartite ranking. This means that advantages gained from solving one task can be carried over to the other task, such as the ability to obtain conditional density estimates, and an order-of-magnitude reduction in computational time for training the algorithm. It also means that some algorithms are robust to the choice of evaluation metric used; they can theoretically perform well when performance is measured either by a misclassification error or by a statistic of the ROC curve (such as the area under the curve). Specifically, we provide such an equivalence relationship between a generalization of Freund et al.'s RankBoost algorithm, called the "P-Norm Push," and a particular cost-sensitive classification algorithm that generalizes AdaBoost, which we call "P-Classification." We discuss and validate the potential benefits of this equivalence relationship, and perform controlled experiments to understand P-Classification's empirical performance. There is no established equivalence relationship for logistic regression and its ranking counterpart, so we introduce a logistic-regression-style algorithm that aims in between classification and ranking, and has promising experimental performance with respect to both tasks.

Keywords: supervised classification, bipartite ranking, area under the curve, rank statistics, boosting, logistic regression

1. Introduction

The success of a machine learning algorithm can be judged in many different ways. Thus, algorithms that are somehow robust to multiple performance metrics may be more generally useful for a wide variety of problems. Experimental evaluations of machine learning algorithms tend to reflect this by considering several different measures of success (see, for example, the study of Caruana and Niculescu-Mizil, 2006). For instance, classification algorithms are commonly judged both by their classification accuracy and the Area Under the ROC curve (AUC), even though these algorithms are designed only to optimize the classification accuracy, and not the AUC.

If algorithms should be judged using multiple measures of success, it makes sense to analyze and design algorithms that achieve success with respect to multiple performance metrics. This is the topic considered in this work, and we show that several additional advantages, such as probabilistic interpretability and computational speed, can be gained from finding equivalence relationships between algorithms for different problems. It is true that there is no "free lunch" (Wolpert and

Macready, 1997), we are not claiming that one algorithm can solve all problems. We instead point out that it is possible to optimize two objectives simultaneously if they have an overlapping set of optima. The objectives in this case are convexified versions of the performance metrics for classification and ranking. In this work, we show that a particular equivalence relationship exists between two algorithms: a ranking algorithm called the P-Norm Push (Rudin, 2009), which is a generalization of RankBoost (Freund et al., 2003) that aims to maximize a weighted area under the curve, and a classification algorithm that we call P-Classification, which is a generalization of AdaBoost (Freund and Schapire, 1997) that aims to minimize a weighted misclassification error. Specifically, we show that P-Classification not only optimizes its objective, but also optimizes the objective of the P-Norm Push (and vice versa, the P-Norm Push can be made to optimize P-Classification's objective function). Thus, P-Classification and the P-Norm Push perform equally well on both of their objectives; P-Classification can be used as a ranking algorithm, and the P-Norm Push can be made into a classification algorithm. This equivalence relationship allows us to: 1) obtain conditional density estimates for $P(y = 1|x)$ for the P-Norm Push (and thus RankBoost as a special case), 2) obtain solutions of the P-Norm Push an order of magnitude faster without sacrificing the quality of the solution at all, and 3) show a relationship between the P-Norm Push's objective and the "precision" metric used in Information Retrieval. This relationship will allow us to conduct a set of controlled experiments to better understand P-Classification's empirical performance. It is not clear that such an equivalence relationship holds between logistic regression and its ranking counterpart; in fact, our experiments indicate that no such relationship can hold.

Bipartite ranking problems are similar to but distinct from binary classification problems. In both bipartite ranking and classification problems, the learner is given a training set of examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ consisting of instances $x \in \mathcal{X}$ that are either positive ($y = 1$) or negative ($y = -1$). The goal of bipartite ranking is to learn a real-valued ranking function $f : \mathcal{X} \to \mathbb{R}$ that ranks future positive instances higher than negative ones. Bipartite ranking algorithms optimize rank statistics, such as the AUC. There is no decision boundary, and the absolute scores of the examples do not matter, instead the values of the scores relative to each other are important. Classification algorithms optimize a misclassification error, and are they are not designed to optimize rank statistics. The "equivalence" is where a classification (or ranking) algorithm aims to optimize not only a misclassification error, but also a rank statistic.

The first work that suggested such equivalence relationships could hold is that of Rudin and Schapire (2009), showing that AdaBoost is equivalent to RankBoost. They showed that AdaBoost, which iteratively minimizes the exponential misclassification loss, also iteratively minimizes RankBoost's exponential ranking loss, and vice versa, that RankBoost with trivial modifications can be made to minimize the exponential misclassification loss. The first result of our work, provided in Section 3, is to broaden that proof to handle more general ranking losses and classification losses. The more general ranking loss is that of the P-Norm Push, which concentrates on "pushing" negatives away from the top of the ranked list. The more general classification loss, determined mainly by the number of false positives, is minimized by P-Classification, which is introduced formally in Section 3. Also in Section 3, we consider another simple cost-sensitive version of AdaBoost, and show the forward direction of its equivalence to RankBoost; in this case, the cost-sensitive version of AdaBoost minimizes RankBoost's objective no matter what the cost parameter is. In Section 4, we will verify the equivalence relationship empirically and provide evidence that no such relationship holds for logistic regression and its ranking counterpart. In Section 5 we discuss the first two main benefits gained by this equivalence relationship described above, namely obtaining probability esti-

mates, and computing solutions faster. In Section 6 we discuss the relationship of P-Classification's objective to the "precision" metric, and evaluate several parameters influencing the performance of P-Classification. As a result, we are able to suggest improvements to boost performance. Section 7 introduces a new logistic-regression-style algorithm that solves a problem in between classification and ranking, in the hopes of performing better than AdaBoost on both counts. Note that this work does not relate directly to work on reductions (e.g., Balcan et al., 2008), since in this work, the same set of features are used for both the classification and ranking problems without any modification. Another recent work that addresses the use of classification algorithms for solving ranking problems is that of Kotlowski et al. (2011), who show loss and regret bounds on the ranking performance of classifiers. Their analysis is useful when equivalence relationships, such as the ones we show here, do not hold.

2. Definitions

We denote the set of instances with positive labels as $\{x_i\}_{i=1,\ldots,I}$, and the set of instances with negative labels as $\{\tilde{x}_k\}_{i=1,\ldots,K}$, where $x_i, \tilde{x}_k \in \mathcal{X}$. Throughout most of the paper, the subscripts i and k will be used as indices over positive and negative instances, respectively. We assume that (x_i, y_i) are drawn from a joint distribution D on $\mathcal{X} \times \{-1, 1\}$. Our goal is to construct a scoring function $f : \mathcal{X} \to \mathbb{R}$ which gives a real valued score to each instance in $\mathcal{X}$. Let $\mathcal{F}$ denote the hypothesis space that is the class of convex combinations of features $\{h_j\}_{j=1\ldots n}$, where $h_j : \mathcal{X} \to \{-1, 1\}$. The function $f \in \mathcal{F}$ is then defined as a linear combination of the features:

$$f := f_{\boldsymbol{\lambda}} := \sum_j \lambda_j h_j,$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ will be chosen to minimize (or approximately minimize) an objective function. We always include a y-intercept (feature that is 1 for all x), assigned to the index $\bar{j}$. This y-intercept term is important in the equivalence proof; in order to turn the P-Norm Push into a classification algorithm, we need to place a decision boundary by adjusting the y-intercept.

In bipartite ranking, the goal is to rank positive instances higher than the negative ones. The quality of a ranking function is often measured by the area under the ROC curve (AUC). The associated misranking loss, related to $1 - \text{AUC}$, is the number of positive instances that are ranked below negative instances:

$$\text{standard misranking loss } (f) = \sum_{i=1}^{I} \sum_{k=1}^{K} \mathbb{1}_{[f(x_i) \leq f(\tilde{x}_k)]}. \tag{1}$$

The ranking loss is zero when all negative instances are ranked below the positives instances.

In binary classification, the goal is to correctly predict the true labels of positive and negative examples. The loss is measured by the misclassification error:

$$\text{standard misclassification loss } (f) = \sum_{i=1}^{I} \mathbb{1}_{[f(x_i) \leq 0]} + \sum_{k=1}^{K} \mathbb{1}_{[f(\tilde{x}_k) \geq 0]}. \tag{2}$$

Since it is difficult to minimize (1) and (2) directly, a widely used approach is to minimize a convex upper bound on the loss. The exponential loss that is iteratively minimized by AdaBoost (Freund

and Schapire, 1997) is one such example:

$$\mathcal{R}^{AB}(\boldsymbol{\lambda}) := \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} + \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)} =: \mathcal{R}_+^{AB}(\boldsymbol{\lambda}) + \mathcal{R}_-^{AB}(\boldsymbol{\lambda}). \tag{3}$$

The ranking counterpart of AdaBoost is RankBoost (Freund et al., 2003). RankBoost's objective function is a sum of exponentiated differences in scores, a convexified version of (1):

$$\mathcal{R}^{RB}(\boldsymbol{\lambda}) := \sum_{i=1}^{I} \sum_{k=1}^{K} e^{-(f_{\boldsymbol{\lambda}}(x_i)-f_{\boldsymbol{\lambda}}(\tilde{x}_k))} = \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)} = \mathcal{R}_+^{AB}(\boldsymbol{\lambda}) \mathcal{R}_-^{AB}(\boldsymbol{\lambda}). \tag{4}$$

A generalization of RankBoost that is considered in this paper is the P-Norm Push (Rudin, 2009), which minimizes the following objective:

$$\mathcal{R}^{PN}(\boldsymbol{\lambda}) := \sum_{k=1}^{K} \left(\sum_{i=1}^{I} e^{-(f_{\boldsymbol{\lambda}}(x_i)-f_{\boldsymbol{\lambda}}(\tilde{x}_k))} \right)^p.$$

By increasing p, one changes how hard the algorithm concentrates on "pushing" high scoring negative examples down from the top of the list. The power p acts as a soft maximum for the highest scoring negative instance. When $p = 1$, P-Norm Push's objective reduces to RankBoost's objective.

We also investigate the equivalence relationship between logistic regression and its ranking counterpart, which we call "Logistic Regression Ranking" (LRR). Logistic regression minimizes the objective:

$$\mathcal{R}^{LR}(\boldsymbol{\lambda}) := \sum_{i=1}^{I} \log\left(1 + e^{-f_{\boldsymbol{\lambda}}(x_i)}\right) + \sum_{k=1}^{K} \log\left(1 + e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)}\right), \tag{5}$$

whereas LRR is defined with the following objective:

$$\mathcal{R}^{LRR}(\boldsymbol{\lambda}) := \sum_{i=1}^{I} \sum_{k=1}^{K} \log\left(1 + e^{-(f_{\boldsymbol{\lambda}}(x_i)-f_{\boldsymbol{\lambda}}(\tilde{x}_k))}\right). \tag{6}$$

LRR bears a strong resemblance to the algorithm RankNet (Burges et al., 2005) in that its objective (6) is similar to the second term of RankNet's objective (Equation 3 in Burges et al., 2005). LRR's objective (6) is an upper bound on the 0-1 ranking loss in (1), using the logistic loss $\log(1 + e^{-z})$ to upper bound the 0-1 loss $\mathbb{1}_{z \leq 0}$.

3. Equivalence Relationships

We now introduce P-Classification, which is a boosting-style algorithm that minimizes a weighted misclassification error. Like the P-Norm Push, it concentrates on "pushing" the negative examples down from the top of the list. Unlike the P-Norm Push, it minimizes a weighted misclassification error (rather than a weighted misranking error), though we will show that minimization of either objective yields an equally good result for either problem. P-Classification minimizes the following loss:

$$\mathcal{R}^{PC}(\boldsymbol{\lambda}) := \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} + \frac{1}{p} \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)p} =: \mathcal{R}_+^{PC}(\boldsymbol{\lambda}) + \mathcal{R}_-^{PC}(\boldsymbol{\lambda}) \tag{7}$$

where $f_{\boldsymbol{\lambda}} = \sum_j \lambda_j h_j$. P-Classification is a generalization of AdaBoost, in that when $p = 1$ (i.e., no emphasis on the top-scoring negatives), P-Classification's loss reduces exactly to that of Ada-Boost's. We implemented P-Classification as coordinate descent (functional gradient descent) on $\mathcal{R}^{PC}(\boldsymbol{\lambda})$. Pseudocode is presented in Figure 1, using the notation of Collins et al. (2002), where i is the index over all examples (not just positive examples), and $\mathbf{M}$ is the "game matrix" for Ada-Boost, where $M_{ij} = y_i h_j(x_i)$. AdaBoost was originally shown to be a coordinate descent algorithm by Breiman (1997), Friedman et al. (2000), Rätsch et al. (2001), Duffy and Helmbold (1999) and Mason et al. (2000).

1. **Input:** examples $\{(x_i, y_i)\}_{i=1}^m$, where $(x_i, y_i) \in \mathcal{X} \times \{-1, 1\}$, features $\{h_j\}_{j=1}^n$, $h_j : \mathcal{X} \to \mathbb{R}$, number of iterations $t_{\max}$, parameter p.

2. **Define:** $M_{ij} := y_i h_j(x_i)$ for all i, j,
   $\phi_i := \mathbb{1}_{\{y_i=1\}} + p\mathbb{1}_{\{y_i=-1\}}$ for all i.

3. **Initialize:** $\lambda_{1,j} = 0$ for $j = 1, \ldots, n$, $\quad d_{1,i} = 1/m$ for $i = 1, \ldots, m$.

4. **Loop for** $t = 1, \ldots, t_{\max}$

   (a) $j_t \in \mathrm{argmax}_j \sum_{i=1}^m d_{t,i} M_{ij}$

   (b) Perform a linesearch for α_t. That is, find a value α_t that minimizes:

   $$\left(\sum_{i=1}^m d_{t,i} M_{i,j_t} e^{-\alpha_t \phi_i M_{i,j_t}} \right)^2$$

   (c) $\lambda_{t+1,j} = \lambda_{t,j} + \alpha_t \mathbb{1}_{j=j_t}$

   (d) $d_{t+1,i} = d_{t,i} e^{-\alpha_t \phi_i M_{i,j_t}}$ for $i = 1, \ldots, m$

   (e) $d_{t+1,i} = \frac{d_{t+1,i}}{\sum_i d_{t+1,i}}$

5. **Output:** $\lambda_{t_{\max}}$

Figure 1: Pseudocode for the P-Classification algorithm.

There are other cost-sensitive boosting algorithms similar to P-Classification. Sun et al. (2007) introduced three "modifications" of AdaBoost's weight update scheme, in order to make it cost sensitive. Modifications I and II incorporate an arbitrary constant (a cost item) for each example somewhere within the update scheme, where the third modification is a blend of Modifications I and II. Since Sun et al. (2007) consider the iteration scheme itself, some of their modifications are not easy to interpret with respect to a global objective, particularly Modification II (and thus III). Although no global objective is provided explicitly, Modification I seems to correspond to approximate minimization of the following objective: $\left[\sum_i \frac{1}{C_i} e^{-(\mathbf{M}\boldsymbol{\lambda}_i)C_i} \right]$. In that sense, P-Classification is almost a special case of Modification I, where in our case, we would take their arbitrary C_i to be assigned the value of ϕ_i. The step size α_t within Modification I is an approximate solution to a line-search, whereas we use a numerical (exact) linesearch. The AdaCost algorithm of Fan et al. (1999)

is another cost-sensitive variation of AdaBoost, however it is not associated with a global objective and in our experiments (not shown here) it tended to choose the intercept repeatedly as the weak classifier, and thus the combined classifier was also the intercept. Lozano and Abe (2008) also use ℓ_p norms within a boosting-style algorithm, but for the problem of label ranking instead of example ranking. However, their objective is totally different than ours; for instance, it does not correspond directly to a 0-1 misranking loss like (1).

We will now show that the P-Norm Push is equivalent to P-Classification, meaning that minimizers of P-Classification's objective are also minimizers of the P-Norm Push's objective, and that there is a trivial transformation of the P-Norm Push's output that will minimize P-Classification's objective. This trivial transformation simply puts a decision boundary in the right place. This proof will generalize the result of Rudin and Schapire (2009), but with a simpler proof strategy; since there are pathological cases where minimizers of the objectives occur only at infinity, the result of Rudin and Schapire (2009) used a Bregman distance technique to incorporate these points at infinity. Our proof does not handle the cases at infinity, but does handle, in a much simpler way, all other cases. These points at infinity occur because the objectives $\mathcal{R}^{PC}$ and $\mathcal{R}^{PN}$ are not strictly convex (though they are convex). For AdaBoost, there is work showing that the minimization problem can be essentially split into two subproblems, one which handles examples near the decision boundary and is strictly convex, and the other which handles examples that become infinitely far away from the boundary (Mukherjee et al., 2011).

The forward direction of the equivalence relationship is as follows:

Theorem 1 *(P-Classification minimizes P-Norm Push's objective)*
If $\boldsymbol{\lambda}^{PC} \in \operatorname{argmin}_{\boldsymbol{\lambda}} \mathcal{R}^{PC}(\boldsymbol{\lambda})$ (assuming minimizers exist), then $\boldsymbol{\lambda}^{PC} \in \operatorname{argmin}_{\boldsymbol{\lambda}} \mathcal{R}^{PN}(\boldsymbol{\lambda})$.

The corresponding proof within Rudin and Schapire (2009) used four main steps, which we follow also in our proof: 1) characterizing the conditions to be at a minimizer of the classification objective function, 2) using those conditions to develop a "skew" condition on the classes, 3) characterizing the conditions to be at a minimizer of the ranking objective function and 4) plugging the skew condition into the equations arising from step 3, and simplifying to show that a minimizer of the classification objective is also a minimizer of the ranking objective.

Proof Step 1 is to characterize the conditions to be at a minimizer of the classification loss. Define $\boldsymbol{\lambda}^{PC}$ to be a minimizer of $\mathcal{R}^{PC}$ (assuming minimizers exist). At $\boldsymbol{\lambda}^{PC}$ we have: for all $j = 1, \ldots, n$:

$$
\begin{aligned}
0 = \frac{\partial \mathcal{R}^{PC}(\boldsymbol{\lambda})}{\partial \lambda_j}\bigg|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{PC}} &= \sum_{k=1}^{K} e^{p \sum_j \lambda_j^{PC} h_j(\tilde{x}_k)} h_j(\tilde{x}_k) + \sum_{i=1}^{I} e^{-\sum_j \lambda_j^{PC} h_j(x_i)} (-h_j(x_i)) \\
&= \sum_{k=1}^{K} v_k^p h_j(\tilde{x}_k) + \sum_{i=1}^{I} q_i (-h_j(x_i)) \quad (8)
\end{aligned}
$$

where $v_k := e^{f_{\boldsymbol{\lambda}^{PC}}(\tilde{x}_k)}$ and $q_i := e^{-f_{\boldsymbol{\lambda}^{PC}}(x_i)}$.

Step 2 is to develop a skew condition on the classes. When j is $\bar{j}$ then $h_j(x_i) = 1$ for all i, and $h_j(\tilde{x}_k) = 1$ for all k. Using this, we can derive the skew condition:

$$
0 = \sum_{k=1}^{K} v_k^p - \sum_{i=1}^{I} q_i := p \mathcal{R}_-^{PC}(\boldsymbol{\lambda}^{PC}) - \mathcal{R}_+^{PC}(\boldsymbol{\lambda}^{PC}). \quad \text{(skew condition)} \quad (9)
$$

Step 3 is to characterize the conditions to be at a minimizer of the ranking loss. First we simplify the derivatives:

$$
\begin{aligned}
\frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j} &= \sum_{k=1}^{K} p \left(\sum_{i=1}^{I} e^{(-f_{\boldsymbol{\lambda}}(x_i) - f_{\boldsymbol{\lambda}}(\tilde{x}_k))} \right)^{p-1} \left[\sum_{i=1}^{I} e^{-(f_{\boldsymbol{\lambda}}(x_i) - f_{\boldsymbol{\lambda}}(\tilde{x}_k))} \left[-(h_j(x_i) - h_j(\tilde{x}_k)) \right] \right] \\
&= p \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)p} \left(\sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} \right)^{p-1} \left(h_j(\tilde{x}_k) \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} - \sum_{i=1}^{I} h_j(x_i) e^{-f_{\boldsymbol{\lambda}}(x_i)} \right) \\
&= p \left(\sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} \right)^{p-1} \left[\sum_{k=1}^{K} h_j(\tilde{x}_k) e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)p} \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} \right. \\
&\qquad\qquad \left. - \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)p} \sum_{i=1}^{I} h_j(x_i) e^{-f_{\boldsymbol{\lambda}}(x_i)} \right]. \quad (10)
\end{aligned}
$$

To be at a minimizer, the derivatives above must all be zero. Continuing to step 4, when $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{PC}$, we have:

$$
\frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j} \bigg|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{PC}} = p \left(\sum_{i=1}^{I} q_i \right)^{p-1} \left[\sum_{k=1}^{K} h_j(\tilde{x}_k) v_k^p \sum_{i=1}^{I} q_i - \sum_{k=1}^{K} v_k^p \sum_{i=1}^{I} h_j(x_i) q_i \right].
$$

Using the skew condition (9), and then (8),

$$
\begin{aligned}
\frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j} \bigg|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{PC}} &= p \left(\sum_{i=1}^{I} q_i \right)^{p-1} \left(\sum_{i=1}^{I} q_i \right) \left[\sum_{k=1}^{K} h_j(\tilde{x}_k) v_k^p - \sum_{i=1}^{I} h_j(x_i) q_i \right] \\
&= p \left(\sum_{i=1}^{I} q_i \right)^{p} \frac{\partial \mathcal{R}^{PC}(\boldsymbol{\lambda})}{\partial \lambda_j} \bigg|_{\boldsymbol{\lambda} = \boldsymbol{\lambda}^{PC}} = 0. \quad (11)
\end{aligned}
$$

This means that $\boldsymbol{\lambda}^{PC}$ is a minimizer of the P-Norm Push's objective. ∎

The backwards direction of the equivalence relationship is as follows:

Theorem 2 *(The P-Norm Push can be trivially altered to minimize P-Classification's objective.) Let $\bar{j}$ index the constant feature $h_{\bar{j}}(x) = 1 \;\; \forall x$. Take $\boldsymbol{\lambda}^{PN} \in \arg\min_{\boldsymbol{\lambda}} \mathcal{R}^{PN}(\boldsymbol{\lambda})$ (assuming minimizers exist). Create a "corrected" $\boldsymbol{\lambda}^{PN,\mathrm{corr}}$ as follows:*

$$
\boldsymbol{\lambda}^{PN,\mathrm{corr}} = \boldsymbol{\lambda}^{PN} + b \cdot \mathbf{e}_{\bar{j}},
$$

where

$$
b = \frac{1}{p+1} \ln \frac{\sum_i e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)}}{\sum_k e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p}}. \quad (12)
$$

Then, $\boldsymbol{\lambda}^{PN,\mathrm{corr}} \in \arg\min_{\boldsymbol{\lambda}} \mathcal{R}^{PC}(\boldsymbol{\lambda})$.

Rudin and Schapire (2009) have a very simple proof for the reverse direction, but this technique could not easily be applied here. We have instead used the following proof outline: first, we

show that the corrected vector $\boldsymbol{\lambda}^{PN,\mathrm{corr}}$ satisfies the skew condition; then by deriving an expression similar to (11), we show that if the corrected P-Norm Push's derivatives are zero, then so are P-Classification's derivatives at the corrected P-Norm Push's solution.

Proof We will first show that the skew condition is satisfied for corrected vector $\boldsymbol{\lambda}^{PN,\mathrm{corr}}$. The condition we need to prove is:

$$\sum_{i=1}^{I} q_i^{\mathrm{corr}} = \sum_{k=1}^{K} (v_k^{\mathrm{corr}})^p \qquad \text{(skew condition)}, \tag{13}$$

where $v_k^{\mathrm{corr}} := e^{f_{\boldsymbol{\lambda}^{PN,\mathrm{corr}}}(\tilde{x}_k)}$ and $q_i^{\mathrm{corr}} := e^{-f_{\boldsymbol{\lambda}^{PN,\mathrm{corr}}}(x_i)}$.

From (12), we have:

$$e^{-b} = \left[\frac{\sum_k e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p}}{\sum_i e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)}} \right]^{\frac{1}{p+1}}.$$

The left side of (13) thus reduces as follows:

$$\sum_{i=1}^{I} q_i^{\mathrm{corr}} = \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)-b} = e^{-b} \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)}$$

$$= \left[\frac{\sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p}}{\sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)}} \right]^{\frac{1}{p+1}} \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)}$$

$$= \left[\sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)} \right]^{\frac{p}{p+1}} \left[\sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p} \right]^{\frac{1}{p+1}}. \tag{14}$$

Now consider the right side:

$$\sum_{k=1}^{K} (v_k^{\mathrm{corr}})^p = \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p} e^{bp}$$

$$= \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p} \left[\frac{\sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)}}{\sum_k e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p}} \right]^{\frac{p}{p+1}}$$

$$= \left[\sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}^{PN}}(x_i)} \right]^{\frac{p}{p+1}} \left[\sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}^{PN}}(\tilde{x}_k)p} \right]^{\frac{1}{p+1}}. \tag{15}$$

Expression (14) is equal to expression (15), so the skew condition in (13) holds. According to (10), at $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{PN,\mathrm{corr}}$,

$$\left. \frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j} \right|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{PN,\mathrm{corr}}} = p \left(\sum_i q_i^{\mathrm{corr}} \right)^{p-1} \left[\sum_k h_j(\tilde{x}_k)(v_k^{\mathrm{corr}})^p \sum_i q_i^{\mathrm{corr}} \right.$$

$$\left. - \sum_k (v_k^{\mathrm{corr}})^p \sum_i h_j(x_i) q_i^{\mathrm{corr}} \right].$$

Incorporating (13),

$$\left. \frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j} \right|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{PN,\mathrm{corr}}} = p \left(\sum_i q_i^{\mathrm{corr}} \right)^{p} \left[\sum_k h_j(\tilde{x}_k)(v_k^{\mathrm{corr}})^p - \sum_i h_j(x_i) q_i^{\mathrm{corr}} \right],$$

which includes the derivatives of $\mathcal{R}^{PC}$:

$$\frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j}\bigg|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{PN,\text{corr}}} = p\left(\sum_i q_i^{\text{corr}}\right)^p \left[\frac{\partial \mathcal{R}^{PC}(\boldsymbol{\lambda})}{\partial \lambda_j}\bigg|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{PN,\text{corr}}}\right].$$

By our assumption that $\boldsymbol{\lambda}^{PN,\text{corr}}$ exists, $\sum_i q_i^{\text{corr}}$ is positive and finite. Thus, whenever $\frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j}\big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{PN,\text{corr}}} = 0$ for all j we have

$$\frac{\partial \mathcal{R}^{PC}(\boldsymbol{\lambda})}{\partial \lambda_j}\bigg|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{PN,\text{corr}}} = 0.$$

We need only that $\frac{\partial \mathcal{R}^{PN}(\boldsymbol{\lambda})}{\partial \lambda_j}\big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{PN,\text{corr}}} = 0, \quad \forall j$. This is not difficult to show, since the correction b never influences the value of $\mathcal{R}^{PN}$, that is, $\mathcal{R}^{PN}(\boldsymbol{\lambda}) = \mathcal{R}^{PN}(\boldsymbol{\lambda}^{\text{corr}})$. ∎

As an alternative to P-Classification, we consider a simple weighted version of AdaBoost. The objective for this algorithm, which we call "Cost-Sensitive AdaBoost," is a weighted combination of $\mathcal{R}_-^{AB}$ and $\mathcal{R}_+^{AB}$. The objective is:

$$\mathcal{R}^{CSA}(\boldsymbol{\lambda}) := \sum_{i=1}^I e^{-f_{\boldsymbol{\lambda}}(x_i)} + C\sum_{k=1}^K e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)} =: \mathcal{R}_+^{AB}(\boldsymbol{\lambda}) + C\mathcal{R}_-^{AB}(\boldsymbol{\lambda}).$$

Cost-Sensitive AdaBoost can be implemented by using AdaBoost's usual update scheme, where the only change from AdaBoost is the initial weight vector: d_0 is set so that the negatives are each weighted C times as much as the positives.

No matter what C is, we prove that Cost-Sensitive AdaBoost minimizes RankBoost's objective. This indicates that Cost-Sensitive AdaBoost is not fundamentally different than AdaBoost itself. To show this, we prove the forward direction of the equivalence relationship between Cost-Sensitive AdaBoost (for any C) and RankBoost. We did not get this type of result earlier for P-Classification, because P-Classification produces different solutions than AdaBoost (and the P-Norm Push produces different solutions than RankBoost). Here is the forward direction of the equivalence relationship:

Theorem 3 *(Cost-Sensitive AdaBoost minimizes RankBoost's objective.)*
If $\boldsymbol{\lambda}^{CSA} \in \text{argmin}_{\boldsymbol{\lambda}} \mathcal{R}^{CSA}(\boldsymbol{\lambda})$ (assuming minimizers exist), then $\boldsymbol{\lambda}^{CSA} \in \text{argmin}_{\boldsymbol{\lambda}} \mathcal{R}^{RB}(\boldsymbol{\lambda})$.

The proof follows the same four steps outlined for the proof of Theorem 1.
Proof Define $\boldsymbol{\lambda}^{CSA}$ to be a minimizer of $\mathcal{R}^{CSA}$ (assuming minimizers exist). At $\boldsymbol{\lambda}^{CSA}$ we have:

$$\begin{aligned} 0 = \frac{\partial \mathcal{R}^{CSA}(\boldsymbol{\lambda})}{\partial \lambda_j}\bigg|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{CSA}} &= \frac{\partial \mathcal{R}_+^{AB}(\boldsymbol{\lambda})}{\partial \lambda_j} + C\frac{\partial \mathcal{R}_-^{AB}(\boldsymbol{\lambda})}{\partial \lambda_j} \\ &= \sum_{i=1}^I q_i(-h_j(x_i)) + C\sum_{k=1}^K v_k h_j(\tilde{x}_k) \end{aligned} \quad (16)$$

where $v_k := e^{f_{\boldsymbol{\lambda}^{CSA}}(\tilde{x}_k)}$ and $q_i := e^{-f_{\boldsymbol{\lambda}^{CSA}}(x_i)}$. The next step is to develop a skew condition on the classes. When j is $\bar{j}$ then $h_j(x_i) = 1$ for all i, and $h_j(\tilde{x}_k) = 1$ for all k. Using this, the skew condition can be derived as follows:

$$0 = C \sum_{k=1}^{K} v_k - \sum_{i=1}^{I} q_i := C \mathcal{R}_-^{AB}(\boldsymbol{\lambda}^{CSA}) - \mathcal{R}_+^{AB}(\boldsymbol{\lambda}^{CSA}). \qquad \text{(skew condition)}$$

We now characterize the conditions to be at a minimizer of the ranking loss. We plug the skew condition into the derivatives from RankBoost's objective, which is given in (4):

$$
\begin{aligned}
\frac{\partial \mathcal{R}^{RB}(\boldsymbol{\lambda})}{\partial \lambda_j} &= \sum_{k=1}^{K} h_j(\tilde{x}_k) v_k \sum_{i=1}^{I} q_i - \sum_{k=1}^{K} v_k \sum_{i=1}^{I} h_j(x_i) q_i \\
&= \sum_{k=1}^{K} h_j(\tilde{x}_k) v_k \left[C \sum_{k=1}^{K} v_k \right] - \left[\sum_{k=1}^{K} v_k \right] \sum_{i=1}^{I} h_j(x_i) q_i \\
&= \left[\sum_{k=1}^{K} v_k \right] \left[C \sum_{k=1}^{K} h_j(\tilde{x}_k) v_k - \sum_{i=1}^{I} h_j(x_i) q_i \right].
\end{aligned}
$$

To be at a minimizer, the derivative must be zero. When $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{CSA}$, from (16) we have:

$$\frac{\partial \mathcal{R}^{RB}(\boldsymbol{\lambda})}{\partial \lambda_j}\bigg|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{CSA}} = \left[\sum_{k=1}^{K} v_k \right] \frac{\partial \mathcal{R}^{CSA}(\boldsymbol{\lambda})}{\partial \lambda_j}\bigg|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{CSA}} = 0.$$

This means that $\boldsymbol{\lambda}^{CSA}$ is a minimizer of RankBoost's objective, regardless of the value of C. ∎

4. Verification of Theoretical Results

The previous section presented theoretical results; in this section and in the following sections we demonstrate that these results can have direct implications for empirical practice. The equivalence of P-Classification and P-Norm Push can be observed easily in experiments, both in the special case $p = 1$ (AdaBoost and RankBoost are equivalent, Section 4.2) as well as for their generalizations when $p > 1$ (in Section 4.3). We further investigated whether a similar equivalence property holds for logistic regression and Logistic Regression Ranking ("LRR," defined in Section 2). We present empirical evidence in Section 4.4 suggesting that such an equivalence relationship does not hold between these two algorithms. Both algorithms have been implemented as coordinate descent on their objectives. Coordinate descent was first suggested for logistic regression by Friedman et al. (2000).

4.1 Data Sets

For the experimental evaluation, we used the Letter Recognition, MAGIC, Yeast and Banana data sets obtained from the UCI repository (Frank and Asuncion, 2010). The Letter Recognition data set consists of various statistics computed from black-and-white rectangular pixel displays, which each represent one of the 26 capital-letters of the English alphabet. The learning task is to determine which letter an image represents. The MAGIC data set contains data from the **M**ajor **A**tmospheric **G**amma **I**maging **C**herenkov Telescope project. The goal is to discriminate the statistical signatures of Monte Carlo simulated "gamma" particles from simulated "hadron" particles. The yeast data set is a collection of protein sequences and the goal is to predict cellular localization sites of each

Data Set	# Training examples	# Test examples	# Features
Letter Recognition	1000	4000	15
MAGIC	1000	4000	11
Yeast	500	984	9
Banana	1000	4000	5

Table 1: Sizes of training/test sets used in the experiments. The number of features column represents the total number of features for the data sets, including the intercept.

protein. Banana is an artificial data set with a banana-shaped distribution of two classes, represented by two features.

For MAGIC, Letter Recognition, and Yeast data sets, the weight of each feature $h_j(x_i)$ was quantized into -1 or +1 based on thresholding on $\text{mean}_i h_j(x_i)$. The MAGIC data set was not further pre-processed beyond this, and "hadrons" were used as the positive class. For the Letter Recognition data set, we transformed the data set to two distinct categories, where the letter A represents the positive class and the remaining letters collectively form the negative class. This transformation created a highly imbalanced data set and presented a challenge in our experimental setup for the RankBoost algorithm, which, in its original implementation uses an analytical solution for the line search for α_t at each iteration. In particular, the analytical solution requires that the fraction in the expression for α_t (Equation 2 in Freund et al., 2003) is neither zero nor infinity. To ensure this, each feature h_j in the training set must have at least one positive example where $h_j = 1$ and a negative example where $h_j = -1$, and similarly, the training set should also contain at least one positive example where $h_j = -1$ and a negative example where $h_j = 1$. Our random sampling of the training sets for the Letter Recognition data set did not often satisfy the requirement on the positive examples for "*x2bar*" and "*x-ege*" features; we thus removed these two features. Note that we could not use RankBoost in its original form, since our features are $\{-1,1\}$-valued rather than $\{0,1\}$-valued. We simply rederived Equation 2 in Freund et al. (2003) to accommodate this. For the Yeast data set, from the 10 different classes of localization sites, we used CYT (cytosolic or cytoskeletal) as the positive class and combined the remaining 9 classes as the negative class. We used the 8 numerical features of the Yeast data set and omitted the categorical feature. We increased the number of features of the Banana data set by mapping the original two features $\{x_1, x_2\}$ to a new four-dimensional feature space $\{x_1', x_1'', x_2', x_2''\}$ by thresholding the original feature values at values of -4 and -2. Namely, we used the mapping:

$$x_i' = \begin{cases} +1 & \text{if } x_i > -2 \\ -1 & \text{otherwise} \end{cases} \quad \text{and} \quad x_i'' = \begin{cases} +1 & \text{if } x_i > -4 \\ -1 & \text{otherwise} \end{cases}.$$

for $i = 1, 2$. The experimental results reported in this section are averaged over 10 random and distinct train/test splits. The size of train/test splits for each data set and the number of features are presented in Table 1.

4.2 Equivalence of AdaBoost and RankBoost

Although AdaBoost and RankBoost perform (asymptotically) equally well, it is not immediately clear whether this equivalence would be able to be observed if the algorithm is stopped before the
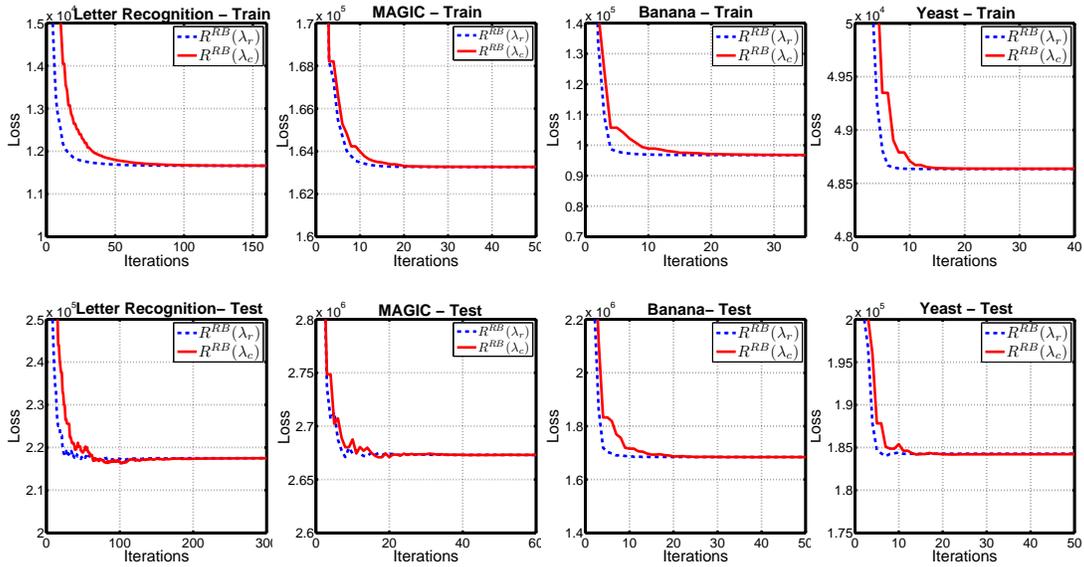
Figure 2: Verifying the forward direction of the equivalence relationship for AdaBoost and Rank-Boost

regime of convergence is reached. We present empirical evidence to support the forward direction of the theoretical equivalence relationship for $p = 1$, on both the training and test splits, for all data sets described above.

In Figure 2, $\{\boldsymbol{\lambda}_r\}_r$ and $\{\boldsymbol{\lambda}_c\}_c$ denote sequences of coefficients produced by RankBoost and Ada-Boost respectively; the subscripts r and c stand for **r**anking and **c**lassification. The figure illustrates both the convergence of AdaBoost and the convergence of RankBoost, with respect to RankBoost's objective R^{RB}. The x-axis denotes the number of iterations. The illustration supports the convergence of AdaBoost to a minimizer of RankBoost's objective. Because of the way that RankBoost is designed, $\mathcal{R}^{RB}(\boldsymbol{\lambda}_r)$ converges more rapidly (in terms of the number of iterations), than $\mathcal{R}^{RB}(\boldsymbol{\lambda}_c)$.

4.3 Equivalence of P-Classification and P-Norm Push

In the same experimental setting, we now validate the equivalence of P-Classification and P-Norm Push. In Figure 3, $\{\boldsymbol{\lambda}_r\}_r$ and $\{\boldsymbol{\lambda}_c\}_c$ denote sequences of coefficients produced by the P-Norm Push and P-Classification respectively. Convergence is illustrated for both algorithms with respect to the P-Norm Push's objective R^{PN}. The x-axis again denotes the number of iterations. The figure illustrates that P-Classification can effectively be used to minimize the P-Norm Push's objective. Comparing with the $p = 1$ results in Figure 2, the convergence behavior on the training sets are similar, whereas there are small differences in convergence behavior on the test sets. One important distinction between training and testing phases is that the ranking loss is required to decrease monotonically on the training set, but not on the test set. As discussed in depth in Rudin (2009), generalization is more difficult as p grows, so we expect a larger difference between training and test behavior for Figure 3.

The next experiment verifies the backwards direction of the equivalence relationship. We demonstrate that a sequence of corrected λ's minimizing P-Norm Push's objective also minimizes P-Classification's objective. At each iteration of the P-Norm Push, we compute b as defined in (12) and update λ_r accordingly. The sequences of $\mathcal{R}^{PC}$ values for $\{\lambda_c\}_c$ and the corrected $\{\lambda_r\}_r$ are shown in Figure 4.

4.4 Equivalence Does Not Seem To Hold For Logistic Regression and Logistic Regression Ranking

We implemented a coordinate descent algorithm for minimizing LRR's objective function (6), where pseudocode is given in Figure 5. Note that the pseudocode for minimizing logistic regression's objective function would be similar, with the only change being that the definition of the matrix M is the same as in Figure 1.

Figure 6 provides evidence that no such equivalence relationship holds for logistic regression and Logistic Regression Ranking. For this experiment, $\{\lambda_r\}_r$ and $\{\lambda_c\}_c$ denote sequences of coefficients produced by LRR and logistic regression respectively. Convergence is illustrated for both algorithms with respect to LRR's objective. Even after many more iterations than the earlier experiments, and after LRR's objective function values have plateaued for the two algorithms, these values are not close together.
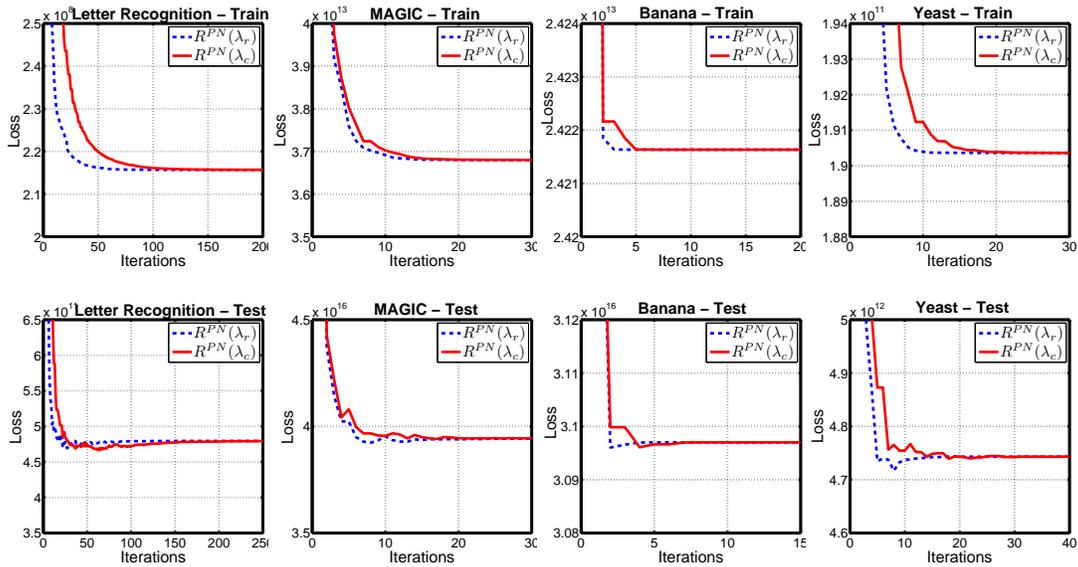

Figure 3: Verifying the forward direction of the equivalence theorem for P-Classification and the P-Norm Push (p=4)

5. Benefits of the Equivalence Relationship

Four major benefits of the equivalence relationship are (i) theoretical motivation, using an algorithm that optimizes classification and ranking objectives simultaneously (ii) gaining the ability to estimate conditional probabilities from a ranking algorithm, (iii) much faster runtimes for ranking tasks, by passing to a classification algorithm and using the equivalence relationship, and (iv) building a relationship between the P-Norm Push and the "precision" performance metric through P-Classification's objective (discussed in Section 6). We have already discussed theoretical motivation, and we will now discuss the other two benefits.

5.1 Estimating Probabilities

The main result in this section is that the scoring function $f_{\boldsymbol{\lambda}}(x)$ can be used to obtain estimates of the conditional probability $P(y = 1|x)$. This result relies on properties of the loss functions, including smoothness, and the equivalence relationship of Theorem 2. Note that the conditional probability estimates for AdaBoost are known not to be very accurate asymptotically in many cases (e.g., see Mease et al., 2007), even though AdaBoost generally provides models with high classification and ranking accuracy (e.g., see Caruana and Niculescu-Mizil, 2006). In other words, even in cases where the probability estimates are not accurate, the relative ordering of probability estimates (the ranking) can be accurate.
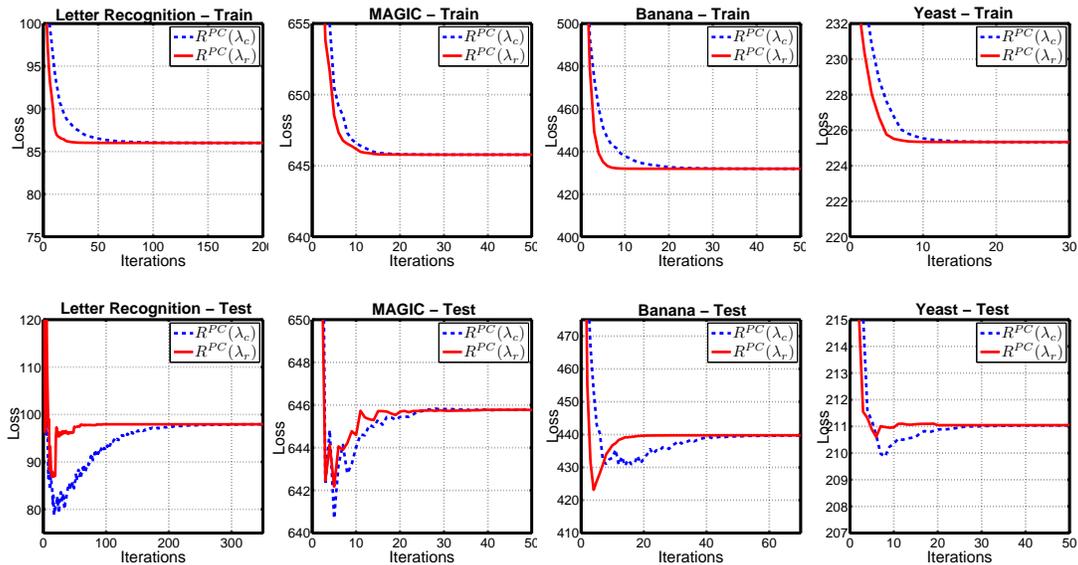

Figure 4: Verifying the backward direction of the equivalence for P-Classification and P-Norm Push (p=4). $\boldsymbol{\lambda}_r$ are corrected with b that is defined in (12).

1. **Input:** Examples $\{(x_i, y_i)\}_{i=1}^m$, where $x_i \in X$, $y_i \in \{-1, 1\}$, features $\{h_j\}_{j=1}^n$, $h_j : X \to \mathcal{R}$, number of iterations $t_{\max}$.

2. **Define:** $M_{ik,j} := h_j(x_i) - h_j(\tilde{x}_k)$ for all i, k, j, where the first index is over all positive-negative pairs indexed by ik, for $ik = 1, \ldots, IK$.

3. **Initialize:** $\lambda_{1,j} = 0$ for $j = 1, \ldots, n$, $d_{1,ik} = 1/IK$ for $ik = 1, \ldots, IK$.

4. **Loop for** $t = 1, \ldots, t_{\max}$

   (a) $j_t \in \mathrm{argmax}_j \sum_{ik} d_{t,ik} M_{ik,j}$

   (b) Perform a linesearch for α_t. That is, find a value α_t that minimizes:

   $$\left(\sum_{ik=1}^{IK} M_{ik,j_t} \frac{1}{1 + e^{\left[\left(\sum_j M_{ik,j_t} \lambda_t \right) + \alpha_{ik} M_{ik,j_t} \right]}} \right)^2$$

   (c) $\lambda_{t+1,j} = \lambda_{t,j} + \alpha_t \mathbb{1}_{j=j_t}$

   (d) $d_{t+1,ik} = \frac{1}{1 + e^{\left(\sum_j M_{ik,j} \lambda_{t+1,j} \right)}}$ for $i = 1, \ldots, I, k = 1, \ldots, K$

   (e) $d_{t+1,ik} = \frac{d_{t+1,ik}}{\sum_{ik} d_{t+1,ik}}$

5. **Output:** $\lambda_{t_{\max}}$

Figure 5: Pseudocode for the Logistic Regression Ranking algorithm.

Theorem 4 *Probability estimates for P-Classification and for the P-Norm Push algorithm (with λ corrected trivially as in Theorem 2) can be obtained from the scoring function $f_\lambda(x)$ as follows:*

$$\hat{P}(y = 1 | x) = \frac{1}{1 + e^{-f_\lambda(x)(1+p)}}.$$

Proof This proof (in some sense) generalizes results from the analysis of AdaBoost and logistic regression (see Friedman et al., 2000; Schapire and Freund, 2011). A general classification objective function can be regarded as an estimate for the expected loss:

$$\mathcal{R}_{\mathrm{true}}(f) := \mathbb{E}_{x,y \sim D}[l(y, f(x))],$$

where expectation is over randomly selected examples from the true distribution $\mathcal{D}$. This quantity can be split into two terms, as follows:

$$\mathbb{E}_{x,y \sim D}[l(y, f(x))] = \mathbb{E}_x \left[\mathbb{E}_{y|x}[l(y, f(x)|x)] \right]$$
$$= \mathbb{E}_x[P(y = 1|x) l(1, f(x)) + (1 - P(y = 1|x)) l(-1, f(x))].$$

At each x, differentiating the inside with respect to $f(x)$ and setting the derivative to 0 at the point $f(x) = f^*(x)$, we obtain:

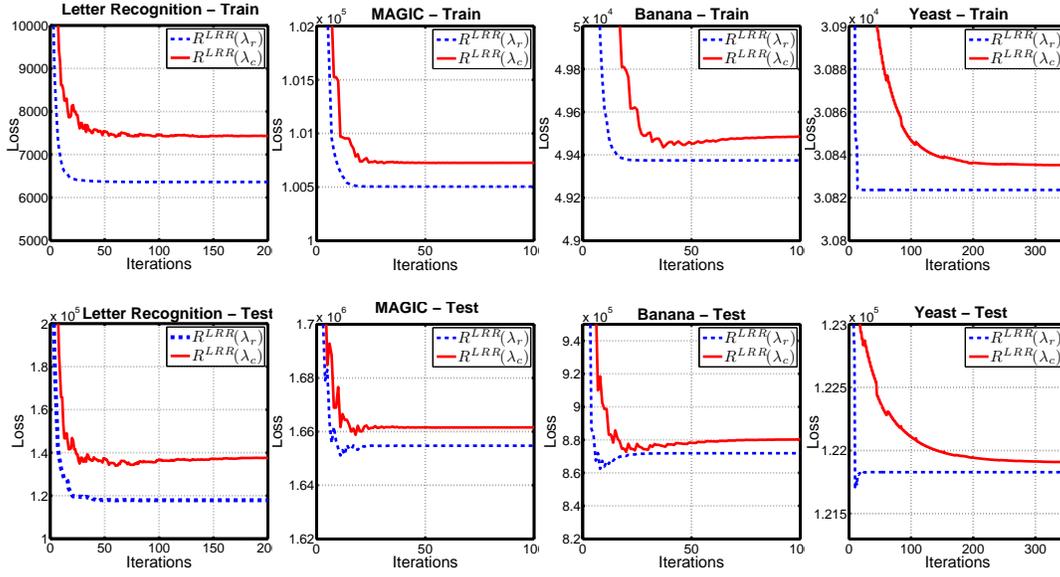$$0 = P(y = 1|x) l'(1, f^*(x)) + (1 - P(y = 1|x)) l'(-1, f^*(x)),$$

Figure 6: Doubt on equivalence relationship for logistic regression and LRR

and solving this equation for $P(y = 1|x)$,

$$P(y = 1|x) = \frac{1}{\frac{l'(1,f^*(x))}{-l'(-1,f^*(x))} + 1}.$$ (17)

P-Classification's objective is an empirical sum over the training instances rather than an expectation over the true distribution:

$$\mathcal{R}^{PC}(\lambda) = \sum_{i=1}^{I} e^{-f_\lambda(x_i)} + \sum_{k=1}^{K} \frac{1}{p} e^{f_\lambda(\tilde{x}_k)p}.$$

Thus, estimates of the conditional probabilities $\hat{P}(y = 1|x)$ can be obtained using (17) where:

$$l(1, f(x)) = e^{-f(x)} \text{ and } l(-1, f(x)) = \frac{1}{p} e^{f(x)p},$$

and instead of $f^*(x)$, which cannot be calculated, we use $f_\lambda(x)$. To do this, we first find derivatives:

$$l(1, f(x)) = e^{-f(x)} \quad \Rightarrow \quad l'(1, f(x)) = -e^{-f(x)}$$

$$l(-1, f(x)) = \frac{1}{p} e^{f(x)p} \quad \Rightarrow \quad l'(-1, f(x)) = e^{f(x)p}.$$

Substituting into (17), we obtain estimated conditional probabilities as follows:

$$\hat{P}(y = 1|x) = \frac{1}{1 + \frac{l'(1,f_\lambda(x))}{-l'(-1,f_\lambda(x))}} = \frac{1}{1 + \frac{-e^{-f_\lambda(x)}}{-e^{f_\lambda(x)p}}} = \frac{1}{1 + e^{-f_\lambda(x)(1+p)}}.$$

This expression was obtained for P-Classification, and extends to the P-Norm Push (with λ corrected) by the equivalence relationship of Theorem 2. ∎

Note that for $p = 1$, Theorem 4 yields conditional probabilities for RankBoost.

Letter Recognition				
	Algorithm	*Objective*	*# of Iterations(for .05%)*	*Time (sec.)*
1K Train	AdaBoost	$\mathcal{R}^{RB}$	123.4±21.6	**0.1**±0.0
	RankBoost	$\mathcal{R}^{RB}$	50.2±10.4	1.1±0.3
	P-Classification	$\mathcal{R}^{PN}$	132.5±23.9	**0.6**±0.1
	PNormPush	$\mathcal{R}^{PN}$	43.3±7.0	3.3±0.6
	Logistic Regression	$\mathcal{R}^{LRR}$	N/A	N/A
	LRR	$\mathcal{R}^{LRR}$	50.0±11.9	8.0±3.0
3K Train	AdaBoost	$\mathcal{R}^{RB}$	112.3±28.5	**0.1**±0.0
	RankBoost	$\mathcal{R}^{RB}$	43.7±9.3	10.7±2.7
	P-Classification	$\mathcal{R}^{PN}$	136.5±36.4	**1.3**±0.3
	PNormPush	$\mathcal{R}^{PN}$	44.0±8.8	29.3±6.5
	Logistic Regression	$\mathcal{R}^{LRR}$	N/A	N/A
	LRR	$\mathcal{R}^{LRR}$	43.8±11.4	65.8±18.2
5K Train	AdaBoost	$\mathcal{R}^{RB}$	108.9±18.8	**0.1**±0.0
	RankBoost	$\mathcal{R}^{RB}$	43.2±7.5	29.2±7.8
	P-Classification	$\mathcal{R}^{PN}$	138.3±29.4	**1.7**±0.3
	PNormPush	$\mathcal{R}^{PN}$	41.9±6.2	72.0±12.5
	Logistic Regression	$\mathcal{R}^{LRR}$	N/A	N/A
	LRR	$\mathcal{R}^{LRR}$	44.4±8.3	218.3±40.6

Table 2: Comparison of runtime performances over varying training set sizes. $p = 4$ for P-Classification and P-Norm Push.

5.2 Runtime Performances

Faster runtime is a major practical benefit of the equivalence relationship proved in Section 3. As we have shown in Sections 4.2 and 4.3, when comparing how ranking algorithms and classification algorithms approach the minimizers of the misranking error, the ranking algorithms tend to converge more rapidly in terms of the number of iterations. Convergence with fewer iterations, however, does not translate into *faster* convergence. Each iteration of either algorithm requires a search for the optimal weak hypothesis j. For the P-Norm Push, each comparison requires quadratic space (involving a vector multiplication of size $I \times K$). In contrast, P-Classification's comparisons are linear in the number of examples (involving a vector multiplication of size $I + K$). For RankBoost, note that a more efficient implementation is possible for bipartite ranking (see Section 3.2 of Freund et al., 2003), though a more efficient implementation has not previously been explored in general for the P-Norm Push; in fact, the equivalence relationship allows us to use P-Classification instead, making it somewhat redundant to derive one.

Table 2 presents the number of iterations and the amount of time required to train each of the algorithms (AdaBoost, RankBoost, P-Classification for p=4, P-Norm Push for p=4, logistic regression, and Logistic Regression Ranking) in an experiment, using a 2.53 gHz macbook pro with 4 GB ram. We used the RankBoost algorithm with the "second method" for computing α_t given by Freund et al. (2003), since it is the special case corresponding to the P-Norm Push with

$p = 1$. The results are presented for the Letter Recognition data set, which is the largest data set in our experimental corpus. To assess the scalability of the algorithms, we generated 10 training sets each of sizes 1000 examples, 3000 examples, and 5000 examples (30 total training sets). For each algorithm, we report the mean and variance (over 10 training sets) of the number of iterations and the time elapsed (in seconds) for the ranking loss to be within 0.05% of the asymptotic minimum ranking loss. The asymptotic value was obtained using 200 iterations of the corresponding ranking algorithm (for AdaBoost and RankBoost, we used RankBoost; for P-Classification and the P-Norm Push, we used the P-Norm Push; and for logistic regression and LRR, we used LRR). Note that logistic regression may never converge to within 0.05% of the ranking loss obtained by LRR (there is no established equivalence relationship), so "N/A" has been placed in the table when this occurs.

Comparing the runtime performances of classification and ranking algorithms in Table 2, Ada-Boost and P-Classification yield dramatic improvement over their ranking counterparts. Despite the fact that they require more than double the number of iterations to obtain the same quality of solution, it only takes them a fraction of the time. Further, AdaBoost and P-Classification appear to scale better with the sample size. Going from 1K to 5K, AdaBoost's run time roughly doubles, on average from 0.08 to 0.14 seconds, whereas RankBoost takes over 27 times longer (29.19/1.08 $\approx$ 27). Similarly, P-Classification's run time on the 5K data set is slightly more than twice the run time on the 1K data set, as opposed to approximately 22 times longer (72/3.32 $\approx$ 21.7) for P-Norm Push on the 5K data set. Thus, the equivalence relationship between classification and ranking algorithms enables us to pass the efficiency of classification algorithms to their ranking counterparts, which leads to significant speed improvement for ranking tasks.

6. Experiments on Prediction Performance

When evaluating the prediction performance of the P-Classification algorithm, we chose *precision* as our performance metric, motivated by a specific relationship between precision and P-Classification's objective that we derive in this section. In Information Retrieval (IR) contexts, precision is defined as the number of relevant instances retrieved as a result of a query, divided by the total number of instances retrieved. Similarly, in a classification task the precision is defined as

$$\text{Precision} := \frac{\text{TP}}{\text{TP} + \text{FP}}$$

where TP (true positives) are the number of instances correctly labeled as belonging to the positive class and FP (false positives) are the number of instances incorrectly labeled as belonging to the positive class. In a classification task, 100% precision means that every instance labeled as belonging to the positive class does indeed belong to the positive class, whereas 0% precision means that all positive instances are misclassified.

In order to derive the relationship between precision and P-Classification's objective, consider P-Classification's objective:

$$\mathcal{R}^{PC}(\boldsymbol{\lambda}) = \sum_{i=1}^{I} e^{-f_{\boldsymbol{\lambda}}(x_i)} + \frac{1}{p} \sum_{k=1}^{K} e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)p}. \tag{18}$$

There is a potential for the second term to be much larger than the first term when $p > 1$, so we consider:

$$\mathcal{R}^{PC}(\boldsymbol{\lambda}) \geq \frac{1}{p} \sum_k e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)p} \tag{19}$$

$$\geq \frac{1}{p} \sum_k e^{\gamma p} \mathbb{1}_{[f_{\boldsymbol{\lambda}}(\tilde{x}_k) \geq \gamma]} = \frac{1}{p} e^{\gamma p} \sum_k \mathbb{1}_{[f_{\boldsymbol{\lambda}}(\tilde{x}_k) \geq \gamma]}. \tag{20}$$

Transitioning from (19) to (20) uses the fact that $e^{f_{\boldsymbol{\lambda}}(\tilde{x}_k)} > e^{\gamma p}$ when $f_{\boldsymbol{\lambda}}(\tilde{x}_k) \geq \gamma, \ \forall \gamma$. Let $\bar{I}_{f_{\boldsymbol{\lambda}} > \gamma}, \bar{K}_{f_{\boldsymbol{\lambda}} > \gamma}$ denote the number of positive and negative instances that score higher than the cutoff threshold γ, respectively. Then,

$$\sum_k \mathbb{1}_{[f_{\boldsymbol{\lambda}}(\tilde{x}_k) \geq \gamma]} = \bar{K}_{f_{\boldsymbol{\lambda}} \geq \gamma}$$

$$= (\bar{I}_{f_{\boldsymbol{\lambda}} \geq \gamma} + \bar{K}_{f_{\boldsymbol{\lambda}} \geq \gamma}) \left(1 - \frac{\bar{I}_{f_{\boldsymbol{\lambda}} \geq \gamma}}{\bar{I}_{f_{\boldsymbol{\lambda}} \geq \gamma} + \bar{K}_{f_{\boldsymbol{\lambda}} \geq \gamma}} \right)$$

$$= (\bar{I}_{f_{\boldsymbol{\lambda}} \geq \gamma} + \bar{K}_{f_{\boldsymbol{\lambda}} \geq \gamma}) (1 - \text{Precision}@(f_{\boldsymbol{\lambda}} = \gamma)). \tag{21}$$

Note that $\bar{I}_{f_{\boldsymbol{\lambda}} \geq \gamma} + \bar{K}_{f_{\boldsymbol{\lambda}} \geq \gamma}$ is simply the number of all instances with scores greater than γ. Plugging (21) into (20) yields

$$\mathcal{R}^{PC}(\lambda) \geq \frac{1}{p} e^{\gamma p} (\bar{I}_{f_{\boldsymbol{\lambda}} \geq \gamma} + \bar{K}_{f_{\boldsymbol{\lambda}} \geq \gamma}) (1 - \text{Precision}@(f_{\boldsymbol{\lambda}} = \gamma))$$

which indicates that minimizing P-Classification's objective may yield solutions that have high precision. Through the equivalence of the P-Norm Push and P-Classification, a similar relationship with precision also exists for the P-Norm Push.

6.1 Effect of p:

In this section, we explore the prediction performance of the P-Classification algorithm with respect to p, for various levels of γ, where γ is the cutoff threshold for calculating precision. We have three hypotheses that we want to investigate, regarding the relationship between p and precision.

Hypothesis 1: The presence of exponent p in the second term in (18) enables P-Classification to achieve improved prediction performance.

The first term of (18) can be much smaller than the second term, due mainly to the presence of p in the exponent. This means that the bound in (19) becomes tighter with the presence of p. This may indicate that the exponent p can influence the algorithm's performance with respect to precision. The empirical analysis that we present later in this section investigates the influence and impact of the exponent p on precision accuracy.

Hypothesis 2: Increasing p in P-Classification's objective yields improved prediction performance.

As p increases, the bound in (19) becomes tighter and the largest terms in $\mathcal{R}^{PC}$ correspond to the highest scoring negative examples. Minimizing $\mathcal{R}^{PC}$ thus "pushes" these negative examples down the ranked list, potentially leading to higher values of precision.

Hypothesis 3: P-Classification can achieve better performance than AdaBoost.

P-Classification is a generalized version of AdaBoost. We hypothesize that as p increases, it will be possible to obtain better prediction performance.

We are able to make the hypotheses above since we have chosen precision to be the performance metric. Another metric for evaluating the performance of a ranking function is *recall*, which is defined as the number of true positives divided by the total number of positive examples. A perfect recall of 100% indicates that all positive examples are above the cutoff threshold. Therefore, if our goal was to optimize recall instead of precision, we would want to put the exponent p on the first term of $\mathcal{R}^{PC}$ rather than the second term, since it will create the effect of pushing the positive examples from bottom to top of the list. As the goal is to concentrate on the correct rankings at the top of the list, we particularly aim at achieving higher precision, rather than higher recall. In many IR systems, including web search, what matters most is how many relevant (positive) results there are on the first page or the first few pages—this is reflected directly by precision. Recall does not accurately represent the performance at the top of the list, since it concerns the performance across *all* of the positives; this would require us to go much farther down the list than is reasonable to consider for these applications, in order to span all of the positive examples.

We will now describe the experimental setup. We chose training sets as follows: for MAGIC, 1000 randomly chosen examples, for Yeast, 500 randomly chosen examples, for Banana, 1000 randomly chosen examples and for Letter Recognition, 1200 examples with 200 positives and 1000 positives to achieve an imbalance ratio of 1:5. Increasing the number of positive examples in the training set of Letter Recognition enabled us to keep the "*x-ege*" attribute, but discard only the "*x2bar*" attribute, due to RankBoost's requirement discussed in Section 4.1. For all data sets except Yeast, we randomly selected 4000 examples as the test set. For Yeast, we selected the remaining 983 examples as the test set. The experiments were conducted for three cutoff thresholds γ, to consider the top 50%, 25% and 10% of the list. The algorithms were run until they had converged (hundreds of iterations).

Table 3 presents the precision results on the test sets from all four data sets. In order to investigate the hypotheses above, we redefine Cost-Sensitive AdaBoost as an algorithm that minimizes the following objective:

$$\mathcal{R}^{AB}_{es}(\boldsymbol{\lambda}) = \sum_{i=1}^{I} e^{-y_i f_\lambda(x_i)} + C\frac{1}{p}\sum_{k=1}^{K} e^{-y_k f_\lambda(\tilde{x}_k)}. \tag{22}$$

In order to test the first hypothesis, we fix $C = 1$. When $C = 1$, (22) resembles P-Classification's objective in (7), the only difference is that p is missing in the exponent. When $C = 1$ and $p = 1$, (22) reduces to AdaBoost's objective (3). In that case, P-Classification and AdaBoost give the same performance trivially. As shown in Table 3, for fixed values of p, where $p > 1$, our experiments indicate that P-Classification yields higher precision than Cost-Sensitive AdaBoost, which agrees with our first hypothesis. To see this, compare element-wise the "AdaB.CS" rows ($C = 1, p > 1$) in the table with the corresponding "P-Classification" rows; this is 36 comparisons that all show P-Classification giving higher precision than AdaBoost. This was a controlled experiment in which the treatment was the presence of the exponent p. Our results indicate that the presence of the exponent p can be highly effective in achieving higher precision values.

In order to test the second hypothesis, we investigated the impact of increasing p in P-Classification's objective. We considered four values of p for each of three cutoff thresholds. Table 3 shows that increasing p in P-Classification's objective leads to higher precision values. To see this, consider the P-Classification rows in Table 3. Within each column of the table, performance improves as p increases. With increasing p, P-Classification focuses more on pushing down the high-scored negative instances from top of the list, yielding higher precision.

		MAGIC			Letter Recognition		
		50%	25%	10%	50%	25%	10%
AdaB.$_{CS}$	p=1,C=1	63.94±3.26	69.40±2.75	94.75±1.03	66.88±8.13	66.88±8.13	72.78±7.20
	p=2,C=1	55.92±2.68	69.40±2.75	94.75±1.03	61.76±8.14	61.76±8.14	72.60±6.99
	p=3,C=1	54.25±1.41	69.40±2.75	94.75±1.03	58.91±7.61	58.91±7.61	72.82±7.24
	p=4,C=1	54.25±1.41	69.40±2.75	94.75±1.03	55.61±7.23	56.02±6.82	72.78±7.20
	p=4,C=$\frac{\#neg}{\#pos}$	55.20±2.35	69.40±2.75	94.75±1.03	68.42±7.09	68.42±7.09	72.88±7.33
P-Class.	p=1,C=1	63.94±3.26	69.40±2.75	94.75±1.03	66.88±8.13	66.88±8.13	72.78±7.20
	p=2,C=1	64.45±3.13	70.22±2.77	94.97±0.97	68.98±8.02	68.98±8.02	77.20±7.62
	p=3,C=1	65.21±2.99	70.46±2.60	95.15±0.91	70.09±7.65	70.09±7.65	78.45±8.63
	p=4,C=1	65.13±2.96	70.60±2.62	95.15±0.94	70.38±7.43	70.38±7.43	78.93±8.41
	p=4,C=$\frac{\#neg}{\#pos}$	82.27±7.68	82.49±7.08	95.15±0.94	90.96±3.39	90.96±3.39	90.96±3.39
LR		65.31±2.90	69.51±2.65	94.78±1.08	69.49±7.63	69.49±7.63	77.10±8.22
LR$_{CS}$	C=$\frac{\#neg}{\#pos}$	84.54±8.64	84.54±8.64	95.00±0.96	90.62±4.18	90.62±4.18	90.62±4.18
		Banana			Yeast		
		50%	25%	10%	50%	25%	10%
AdaB.$_{CS}$	p=1,C=1	75.56±0.39	88.77±1.75	90.35±2.49	52.77±4.59	52.77±4.59	56.27±3.87
	p=2,C=1	75.56±0.39	88.77±1.75	90.35±2.49	44.24±2.54	49.11±3.91	56.06±3.85
	p=3,C=1	74.91±0.94	88.77±1.75	90.35±2.49	42.66±1.56	49.11±3.91	56.06±3.85
	p=4,C=1	74.15±1.09	88.77±1.75	90.35±2.49	42.66±1.56	49.11±3.91	56.06±3.85
	p=4,C=$\frac{\#neg}{\#pos}$	75.56±0.39	88.77±1.75	90.35±2.49	42.66±1.56	49.11±3.91	56.06±3.85
P-Class.	p=1,C=1	75.56±0.39	88.77±1.75	90.35±2.49	52.77±4.59	52.77±4.59	56.27±3.87
	p=2,C=1	75.57±0.39	89.16±0.60	96.70±2.79	53.20±4.23	53.20±4.23	56.37±4.66
	p=3,C=1	75.57±0.39	89.16±0.60	97.80±0.86	53.25±4.22	53.25±4.22	56.47±5.38
	p=4,C=1	75.57±0.39	89.16±0.60	98.55±1.02	53.26±4.22	53.26±4.22	56.98±4.74
	p=4,C=$\frac{\#neg}{\#pos}$	88.75±9.45	92.86±2.86	98.55±1.02	56.37±6.26	56.37±6.26	58.12±6.20
LR		75.57±0.39	89.16±0.60	90.35±2.49	53.23±4.24	53.23±4.24	56.57±5.15
LR$_{CS}$	C=$\frac{\#pos}{\#neg}$	88.75±9.45	92.86±2.86	90.35±2.49	55.88±6.09	55.88±6.09	57.59±6.57

Table 3: Precision values at the top 50%, 25% and 10% of the ranked list.

Evaluating the third hypothesis, Table 3 shows that P-Classification for $p > 1$ yields superior precision than AdaBoost in all comparisons (36 of them in total, from 4 data sets, 3 γ levels and 3 values of $p > 1$).

6.2 Effect of Parameter C

Our next set of experiments focus on the behavior that we observe in Cost-Sensitive AdaBoost's results, which is that increasing p has a detrimental effect on precision, whereas in P-Classification, increasing p leads to higher precision. Given that the only difference between $\mathcal{R}^{PC}$ and $\mathcal{R}^{AB}_{es}$ is the presence of the exponent p in $\mathcal{R}^{PC}$, the behavior that we observe can be explained by the hypothesis that the exponent p in P-Classification is the dominant factor in determining the misclassification penalty on the negative examples, overwhelming the effect of the $\frac{1}{p}$ factor.

This leaves room for the possibility that altering the $\frac{1}{p}$ factor could lead to improved performance. We tested this possibility as follows: first, we varied the coefficient C in Cost-Sensitive AdaBoost's objective in (22); second, we introduced the same C into P-Classification's objective,

and thus defined a Cost-Sensitive P-Classification algorithm that minimizes the following loss:

$$\mathcal{R}_{es}^{PC}(\boldsymbol{\lambda}) = \sum_{i=1}^{I} e^{-f_{\lambda}(x_i)} + C\frac{1}{p}\sum_{k=1}^{K} e^{f_{\lambda}(\tilde{x}_k)p}.$$

In the experiments, we heuristically set $C = \frac{\#neg}{\#pos}$ in order to reduce, and possibly eliminate, the detrimental effect of the $\frac{1}{p}$ term. Our data sets share characteristics similar to many other real world data sets in that the number of positive examples is less than the number of negative examples; therefore $C > 1$ for all four data sets. The $\frac{\#neg}{\#pos}$ ratios averaged over 10 splits for each data set are 354/646, 200/1000, 400/600 and 155/345 for MAGIC, Letter Recognition, Banana and Yeast, respectively. The last row in Table 3 for Cost-Sensitive AdaBoost, and also the last row for P-Classification, contains performance results with $C = \frac{\#neg}{\#pos}$. As seen, for a fixed p ($p = 4$ in this case), using this new value for C dramatically improves precision for P-Classification in most cases (10 out of 12 comparisons) and for AdaBoost in some cases (5 out of 12 comparisons). To see this, compare the $p = 4, C = 1$ row with the $p = 4, C = \frac{\#neg}{\#pos}$ row for each algorithm. Using $C > 1$ is equivalent to giving higher misclassification penalty to the negative examples, which can result in a stronger downward push on these examples, raising precision.

6.3 Comparison of P-Classification With Logistic Regression

Table 3 also presents results for logistic regression, both using its original formulation (5) as well as its cost-sensitive variant; the objective for cost-sensitive logistic regression is formulated by multiplying the second term of logistic regression's objective in (5) with the coefficient C. In comparing P-Classification (p=4) with logistic regression, P-Classification performed worse than logistic regression in only one comparison out of 12 (3 γ levels and 4 data sets). Considering their cost-sensitive variants with $C = \frac{\#neg}{\#pos}$, P-Classification and logistic regression generally outperformed their original (non-cost-sensitive) formulations (10 out of 12 comparisons for P-Classification vs. Cost-Sensitive P-Classification with p=4, and 11 out of 12 comparisons for logistic regression vs cost-sensitive logistic regression). Furthermore, Cost-Sensitive P-Classification performed worse than cost-sensitive logistic regression in only 2 out of 12 comparisons.

7. A Hybrid Approach for Logistic Regression

As we discussed in Section 4.4, logistic regression and LRR do not seem to exhibit the equivalence property that we have established for boosting-style algorithms. Consequently, neither logistic regression or LRR may have the benefit of low classification loss and ranking loss simultaneously. This limitation can be mitigated to some degree, through combining the benefits of logistic regression and LRR into a single hybrid algorithm that aims to solve both classification and ranking problems simultaneously. We define the hybrid loss function as:

$$\mathcal{R}^{LR+LRR} = \mathcal{R}^{LR} + \beta\mathcal{R}^{LRR}$$

where β denotes a non-negative regularization factor. $\beta = 0$ reduces the hybrid loss to that of logistic regression, whereas increasing β tilts the balance towards LRR. The trade-off between classification and ranking accuracy is shown explicitly in Figure 7, which presents the 0-1 classification loss and 0-1 ranking loss of this hybrid approach at various β settings. For comparison, we have included
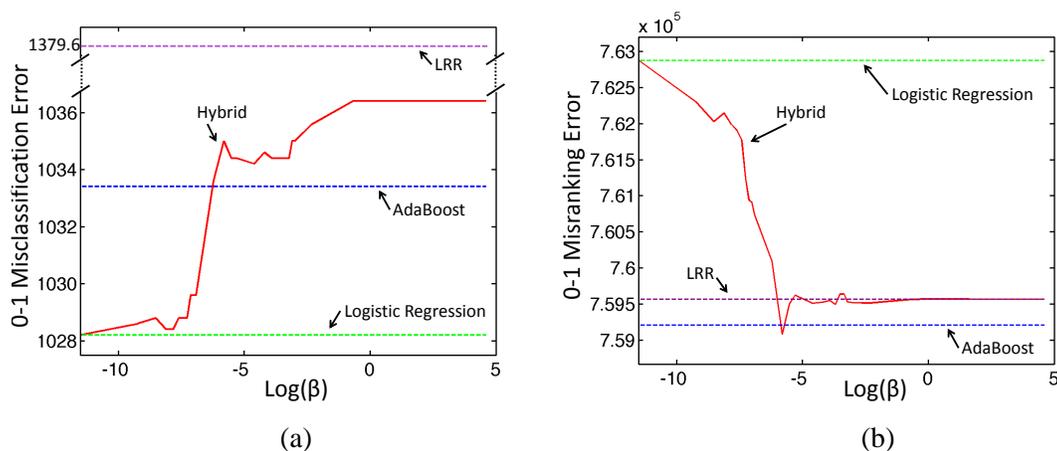
Figure 7: Effect of β on the misclassification error and misranking error rates for MAGIC data set.

the baseline performance of AdaBoost, logistic regression and LRR. For this particular experiment, logistic regression was able to achieve a better misclassification result than AdaBoost (see Figure 7(a)), but at the expense of a very large misranking error (see Figure 7(b)). As β increases, Figure 7(b) shows that the misranking error decreases almost to the level of AdaBoost's, whereas Figure 7(a) shows that the classification error increases to be higher than AdaBoost's. The value of β should be chosen based on the desired performance criteria for the specific application, determining the balance between desired classification vs ranking accuracy.

8. Conclusion

We showed an equivalence relationship between two algorithms for two different tasks, based on a relationship between the minimizers of their objective functions. This equivalence relationship provides an explanation for why these algorithms perform well with respect to multiple evaluation metrics, it allows us to compute conditional probability estimates for ranking algorithms, and permits the solution of ranking problems an order of magnitude faster. The two algorithms studied in this work are generalizations of well-known algorithms AdaBoost and RankBoost. We showed that our new classification algorithm is related to a performance metric used for ranking, and studied empirically how aspects of our new classification algorithm influence ranking performance. This allowed us to suggest improvements to the algorithm in order to boost performance. Finally, we presented a new algorithm inspired by logistic regression that solves a task that is somewhere between classification and ranking, with the goal of providing solutions to both problems. This suggests many avenues for future work. For instance, it may be possible to directly relate either the objective of P-Classification or the P-Norm Push to other performance metrics (see also the discussion in Rudin, 2009). It may also be interesting to vary the derivation of P-Classification to include an exponent on both terms in order to handle, for instance, both precision and recall.

Acknowledgments

References

Maria-Florina Balcan, Nikhil Bansal, Alina Beygelzimer, Don Coppersmith, John Langford, and Gregory B. Sorkin. Robust reductions from ranking to classification. *Machine Learning*, 72: 139–153, 2008.

Leo Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 89–96, 2005.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 161–168, 2006.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48:253–285, 2002.

Nigel Duffy and David P. Helmbold. A geometric approach to leveraging weak learners. In *European Conference on Computational Learning Theory (EuroCOLT)*. Springer-Verlag, 1999.

Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pages 97–105, 1999.

Andrew Frank and Arthur Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

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

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research (JMLR)*, 4:933–969, 2003.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–374, 2000.

Wojciech Kotlowski, Krzysztof Dembczynski, and Eyke Huellermeier. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1113–1120, 2011.

Aurélie C. Lozano and Naoki Abe. Multi-class cost-sensitive boosting with p-norm loss functions. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 506–514, 2008.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2000.

David Mease, Abraham J. Wyner, and Andreas Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research (JMLR)*, 8:409–439, 2007.

Indraneel Mukherjee, Cynthia Rudin, and Robert Schapire. The rate of convergence of AdaBoost. In *Proceedings of the 24th Annual Conference on Learning Theory (COLT)*, 2011.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

Cynthia Rudin. The P-Norm Push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research (JMLR)*, 10:2233–2271, 2009.

Cynthia Rudin and Robert E. Schapire. Margin-based ranking and an equivalence between AdaBoost and RankBoost. *Journal of Machine Learning Research (JMLR)*, 10:2193–2232, 2009.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2011. In Preparation.

Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40:3358–3378, 2007.

David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.